الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur
et de la Recherche Scientifique
Université Akli Mohand Oulhadj - Bouira -
ⵜⴰⵙⴷⴰⵡⵉⵜ ⴰⴽⵍⵉ ⵎⵓⵃⴻⵏⴷ ⵓⵍⵃⴰⵊ - ⵜⵓⴱⵉⵔⴻⵜ -

وزارة التعليم العالي والبحث العلمي
جامعة أكلي محند أولحاج
- البويرة -

Faculté des Sciences et des Sciences Appliquées

كلية العلوم والعلوم التطبيقية

THÈSE

Présentée en vue d'obtenir le grade de

DOCTEUR EN INFORMATIQUE

Spécialité intelligence artificielle et évaluation des performances

Par

Mohamed Yassine HAYI

Vers une gestion intelligente des services sensibles au contexte : Approche centrée utilisateur

---

Directeur de thèse : **Pr. Hamouma MOUMEN**
Co-directeur de thèse : **Dr. Zahira CHOUIREF**

Thèse soutenue le : **16/02/2023**
Devant la commission d'examen composé de :

Pr. Mourad AMAD,	Professeur, Université de Bouira	Président du jury
Pr. Hamouma MOUMEN,	Professeur, Université de Batna 2	Directeur de thèse
Dr. Zahira CHOUIREF,	MCA, Université de Bouira	Co-directeur de thèse
Dr. Chafik ARAR,	MCA, Université de Batna 2	Examinateur
Dr. Bilel SAOUD,	MCA, Université de Bouira	Examinateur
Dr. Aicha AID,	MCB, Université de Bouira	Invité

Laboratoire d'Informatique et de Mathématiques (LIM)

THÈSE

Remerciements

Je remercie, en premier lieu Allah, le Tout puissant, de m'avoir permis et accordé la volonté, la patience et le courage pour réaliser ce travail.

Je tiens à remercier mon directeur de thèse, Monsieur Hamouma MOUMEN, Professeur à l'université de Batna, pour ses conseils et son assistance, ainsi que la co-directrice de thèse, Dr. Zahira CHOUIREF, maître de conférences à l'université de Bouira, pour m'avoir dirigé dans mon travail de recherche tout au long de ma thèse, et m'avoir fait surtout profiter de sa précieuse expérience. Je lui suis reconnaissant pour tout le temps qu'elle m'ait consacré, ainsi que pour ses conseils et son aide précieuse.

J'exprime aussi ma gratitude à Monsieur Mourad AMAD, Professeur à l'université de Bouira pour avoir accepté de présider le jury. Je remercie également les membres du jury, Dr. Chafik ARAR, Dr. Bilel SAOUD et Dr. Aicha AID d'avoir accepté d'évaluer ce travail. Leurs pertinentes remarques et suggestions permettrons surement d'enrichir et d'améliorer notre travail.

Je remercie les enseignants du département d'informatique de l'université de Bouira et de l'université de Médéa, pour leurs conseils et encouragements.

Je remercie chaleureusement toute ma famille, en particulier mes chers parents, qui m'ont apporté leur soutien, ainsi que mes deux frères Abdelmadjid et Mohamed Walid et tous mes amis pour leurs encouragements. Enfin, je tiens à remercier tous ceux qui, de près ou de loin, ont aidé à l'élaboration de ce travail. Encore une fois, merci à tous.

Résumé

De nos jours, la toile contient une quantité très importante et étendue d'information. Profiter de ces larges sélections devient très compliqué pour les utilisateurs qui doivent passer beaucoup de temps à trouver des items qui leurs conviennent et reflètent leurs préférences. L'un des principaux domaines de recherche liés au problème de la surcharge d'informations aujourd'hui est le domaine des systèmes de recommandation. Les systèmes de recommandation sont largement utilisés avec succès dans divers domaines pour fournir de manière automatique des suggestions d'informations pertinentes et personnalisées au demandeur de services selon ses goûts et ses préférences.

L'aspect clé des systèmes de recommandation est l'utilisation des algorithmes d'apprentissage automatique. Ces algorithmes jouent un rôle de plus en plus important car ils déterminent l'information qui serait essentielle et pertinente. L'objectif de notre travail est de proposer un système de recommandation capable de sélectionner les meilleurs items en prenant en considération à la fois les préférences, le profil et le contexte de l'utilisateur, contrairement aux systèmes existants. Le système proposé permet de recommander au demandeur de services des points d'intérêts touristiques sensibles au contexte, en se basant sur les techniques d'optimisation et les techniques d'apprentissage automatique. Ces dernières ont connu un grand intérêt en raison de leur bonne performance sur les grands volumes de données et leur capacité d'analyse et d'extraction d'informations pertinentes.

L'approche proposée est capable d'adapter intelligemment les services selon les besoins des utilisateurs en tenant compte, d'une part, des besoins et des préférences exprimés par les utilisateurs et d'autre part, du contexte environnemental (la situation météorologique et la localisation) et de ses différents changements en temps réel. Cette adaptation est réalisée via deux nouveaux algorithmes hybrides : (i) Le premier algorithme nommé IOGA (Improved Optimization Genetic Algorithm) qui est un algorithme d'optimisation utilisé pour le calcul des k-plus

courts chemins. Le but d'IOGA est de chercher les meilleures routes entre le demandeur de services et les points d'intérêts (la destination) car les items proches ont une chance élevée d'être recommandés par rapport aux items qui sont loin. Pour valider les performances d'IOGA, nous avons créé deux jeux de données, un graphe orienté de 10 000 nœuds et un autre non-orienté de 5 000 nœuds. Ensuite, une comparaison des résultats avec l'algorithme Dijkstra, l'algorithme génétique et l'algorithme A* a été effectué. Les résultats montrent l'efficacité de notre algorithme en termes de courte distance obtenue, du nombre de chemins retournés, du temps d'exécution et de la complexité.

(ii) Le deuxième algorithme appelé H-RN (Hybrid Random Forest Naïve Bayes) qui est une hybridation de deux algorithmes d'apprentissage automatique : classification naïve bayésienne et forêts aléatoires. Ces algorithmes sont très connus et très efficaces dans le domaine de la recommandation.

L'hybridation de ces deux algorithmes permet de surmonter les inconvénients de chacun d'eux pour aboutir à une nouvelle approche intelligente capable de gérer le contexte, de sélectionner et d'adapter les services touristiques en temps réel suivant les attentes de l'utilisateur, sa localisation, l'état de la météo et le degré de température ainsi que l'humidité.

Nous avons implémenté H-RN et quatre autres algorithmes (Naïve Bayes, Random Forest, K-Nearest Neighbor et Neural Networks) sur quatre jeux de données volumineux de différentes tailles. Pour valider notre proposition, nous avons appliqué : plusieurs *critères d'évaluation* tels que : précision, rappel, f-mesure et accuracy avec deux *techniques de validation* (split validation et cross-validation), des *tests statistiques* tels que : One-Way-ANOVA, des *métriques d'erreur* telles que : MAE et RMSE, et *l'étude de complexité*. Les différentes expérimentations sont réalisées *sans et avec* prise en compte des conditions météorologiques afin de montrer l'impact du contexte dans nos travaux de recherche. Les résultats de ces expérimentations ont montré l'efficacité de l'approche proposée. La prise en compte du contexte a amélioré le taux des mesures d'évaluations et l'algorithme H-RN donne de bons résultats en termes de qualité de recommandation, du taux

d'erreur et de complexité.

Mots Clés : Système de recommandation, Apprentissage automatique, Sensibilité au contexte, Profil, Préférences, Optimisation, K-plus court chemins, Tourisme.

Abstract

Nowadays, the web contains a very large and extensive amount of information. Taking advantage of these wide selections becomes very complicated for users who have to spend a lot of time finding items that suit them and reflect their preferences. One of the main areas of research related to the problem of information overload today is the field of recommender systems. Recommender systems are widely and successfully used in various fields to automatically provide suggestions of relevant and personalized information to the requester of services according to their tastes and preferences. The key aspect of recommender systems is the use of machine learning algorithms. These algorithms play an increasingly important role as they determine what information would be essential and relevant.
The objective of our work is to propose a recommender system capable of selecting the best items by taking into consideration both of preferences, the profile and the context of the user, unlike existing systems. The proposed system makes it possible to recommend to the service requester context-sensitive tourist points of interest, based on optimization techniques and machine learning techniques. The latter have been of great interest because of their good performance on large volumes of data and their ability to analyse and extract relevant information.

The proposed approach is able to intelligently adapt the services according to the needs of the users taking into account, on the one hand, the needs and preferences expressed by the users and on the other hand, the environmental context (the meteorological situation and the localization) and its various changes in real time. This adaptation is carried out via two new hybrid algorithms :

(i) The first algorithm named IOGA (Improved Optimization Genetic Algorithm) is an optimization algorithm used for the calculation of the k-shortest paths. The goal of IOGA is to find the best routes between the service requester and the points of interest (the destination) because nearby items have a high chance of being recommended compared to items that are far away. To validate

the performance of IOGA, we created two datasets, a directed graph of 10000 nodes and another undirected of 5000 nodes. Then, a comparison of the results with the Dijkstra algorithm, the genetic algorithm and the A* algorithm was performed. The results show the efficiency of our algorithm in terms of short solutions obtained, number of paths returned, execution time and complexity.

(ii) The second algorithm called H-RN (Hybrid Random Forest Naïve Bayes) is a hybridization of two machine learning algorithms : Naive Bayes and Random Forest. These algorithms are very well known and very effective in the field of the recommendation. The hybridization of these two algorithms makes it possible to overcome the disadvantages of each of them to lead to a new intelligent approach capable of managing the context, selecting and adapting the service in real time according to the expectations of the user, his location, the state of the weather and the degree of temperature as well as the humidity. We implemented H-RN and four other algorithms (Naive Bayes, Random Forest, K-Nearest Neighbour and Neural Networks) on four large datasets of different sizes. To validate our proposal, we applied : several *evaluation criteria* such as : precision, recall, f-measure and accuracy with two *validation techniques* (split validation and cross-validation), *statistical tests* such as : One-Way-ANOVA, *error metrics* such as : MAE and RMSE, and the study of *complexity*. The different experiments were carried out *without and with* taking into account the meteorological conditions in order to show the impact of the context in our research work.
The results of these experiments showed the effectiveness of the proposed approach. Taking the context into account has improved the rate of evaluation measures and that H-RN gives good results in terms of quality of recommendation, error rate and complexity.

Keywords : Recommendation System, Machine learning, Context-awarness, Profile, Preferences, Optimisation, K-shortest path,Tourism.

ملخص

تطور عالم المعلوماتية و الانترنت بطريقة سريعة في العشرية الاخيرة مما ادى لتكون حجم كبير جدا من المعلومات و البيانات الرقمية المخزنة على الشبكة العنكبوتية. هذه الكمية الهائلة تصعب المأمورية امام المستخدمين والذين يحتاجون وقت كبير لإيجاد ضالتهم على حساب تفضيلاتهم و طلباتهم. ولعل افضل طريقة لتسهيل ايجاد البيانات على حسب الحاجة هي استعمال انظمة التوصية والتي ساهمت بشكل كبير في ارضاء المستخدمين.

هدفنا من هذا المشروع هو انشاء نظام توصية جديد يسمح باقتراح مناطق سياحية على حسب ملف, تفضيلات و سياق المستخدم. يعتمد هذا النظام الجديد على تقنيات التحسين والتعلم الالي حيث ان هذه التقنيات الاخيرة معروفة بنجاعتها على البيانات ذات الحجم الكبير.

تعتبر البيانات الرقمية المذكورة سالفا كنزا حقيقيا لما تحتويه من معلومات و احصائيات قد تفيد الانسان بطريقة او بأخرى, ولعل افضل مثال يتمثل في أنظمة التوصية والتي تتيح توجيه المستخدم في أنشطته المتعلقة بالتعلم, السياحة, المشتريات وغيرها من المجالات. التحسين والتوصية هما طريقتان تجعلان حياة الناس أسهل بطريقة مباشرة حيث أن التحسين يوفر الوقت للمستخدمين، والتوصية تسمح للجهاز بالقيام بالاختيار الأفضل بدل الانسان و في مدة زمنية قصيرة. تعتبر تقنيات التحسين واحدة من افضل التقنيات المعروفة في مجال الاعلام الالي. حيث تساعد على القيام بعدة احتمالات في وقت قياسي من أجل إيجاد افضل حل لمشكلة معينة.

لقد قمنا في هذا المشروع بإنشاء خوارزمية هجينة جديدة في مجال التحسين، و قمنا أيضا بإنشاء قاعدة بيانات غير موجهة ذات 5000 مركز، و قاعدة بيانات أخرى موجه ذات 10000 مركز. وقمنا بتطبيق الخوارزمية الجديدة مع ثلاث خورزميات معروفة في المجال على القاعدتين المذكورتين سالفا، نتائج الجزء الأول من مشروعنا اظهرت تفوق الخوارزمية الهجينة على باقي الخوارزميات من حيث نوعية الحلول و عدد الحلول وكذا سرعة الاستجابة.

يتعلق الجزء الثاني بأنظمة التوصية للنقاط السياحية ذات الأهمية باستخدام خوارزمية التحسين الهجينة و المشروحة في الجزء الاول من المشروع. هدفنا هو إنشاء نظام قادر على اختيار أفضل العناصر مع مراعاة خصائص المستخدم وتفضيلاته وسياقه. تقدم أنظمة التوصية الحالية خدمات موثوقة وذات جودة عالية دون مراعاة الأبعاد الثلاثة المذكورة أعلاه. التنوع بين المستخدمين كبير جدًا، والأذواق مختلفة، على سبيل المثال، بين الرجال والنساء، أو بين الصغار والكبار، وحتى بين النهار والليل. لكل منا تفضيلاته الخاصة، وهذا ما يسمى الحساسية للملفات الشخصية والتفضيلات. تتمثل حساسية السياق في نقطتين مهمتين هما حالة الارصاد الجوية و الموقع الجغرافي.

تم تطبيق الخوارزمية الهجينة الثانية على اربع قاعدات بيانية مفتوحة المجال و تم مقارنة النتائج المحصل عليها مع اربع خوارزميات اخرى معروفة في مجال التوصية, النتائج النهائية ابرزت اهمية التهجين بين الخوارزميات و ذلك من خلال العديد من التجارب التي قمنا بها, كما ان استعمال حالة الارصاد الجوية ساهم في تحسين مختلف النتائج.

الكلمات المفتاحية : نظام التوصية ، التعلم الآلي , السياق الشخصي ، الملف الشخصي ، التفصيلات ، التحسين , أقصر المسارات ، السياحة.

Table des Matières

Table des figures

Liste des tableaux

Liste des abréviations

ABCA : Artificial Bee Colony Algorithm

AED : Analyse Exploratoire des Données

ANFIS : Adaptive Neuro Fuzzy Inference System

ANOVA : ANalysis Of VAriance

API : Application Programming Interface

APSP : All Even Shortest Path Algorithms

BDD : Base De Données

BH : Best Humidity

BT : Best Temperature

CA : Clouded Atmospher

CPU : Central Processing Unit

DE : Differential Evolution

DGA : Dijkstra Genetic Algorithm

DisTem : Distance de la Température

DisWea : Distance de la Météo

ET : Endroits Touristiques

FBC : Filtrage Basé sur le Contenu

FC : Filtrage Collaboratif

FCM : Fuzzy C Means

FN : False Negative

FP : False Positive

GA : Genetic Algorithm

GPS : Global Positioning System

HH : High Humidity

HR : Humidité en temps Réel

HR : Heavy Rain

H-RN : Hybrid Random forest Naïve Bayes

HT : High Temperature

IHM : Interface Homme Machine

IOGA : Improved Optimization Genetic Algorithm

IOS : Intersection by Optimizing the Signal

IPD : Intra Package Diversity

JSON : Java Script Object Notation

KNN : k-Nearest Neighbours

LAT : Latitude

LH : Low Humidity

LON : Longitude

LSTM : Long-Short Term Memory

LT : Low Temperature

MAE : Mean Absolute Error

ML : Machine Learning

MWDS : Minimum Weight Doifying Set

NB : Naïve Bayes

NIPD : Nombre Total des Items Pertinent Disponible

NIPS : Nombre des Items Pertinents Sélectionné

NN : Neural Network

NTI : Nombre Total des Items

NTIP : Nombre Total des Items Pertinents

NTIS : Nombre Total des Items Sélectionnée

NTRS : New Travel Recommendation System

PC : Personal Computer

PCC : Pearson Correlation Coefficient

PoI : Points of Interest

R : Rain

RAM : Random Access Memory

RBFN : Radial Basis Function Networks

RF : Random Forest

RI : Recherche d'Information

RMSE : Root Mean Square Error

S : Snow

ScoWea : Score of Weather

SR : Système de Recommandation

SSSP : Single Source Shortest Path Algorithms

SVD : Singular Value Decomposition

SW : Sunny Weather

TE : Temps d'Exécution

TF-IDF : Terme Frequency-Inverse Document Frequency

TN : True Negative

TNR : True Negative Rate

TP : True Positive

TPR : True Positive Rate

TR : Température en temps Réel

TRF : Tree Random Forest

UTC : Universal Time Coordinated

VRSDP : Vehicle Routing issue with Simultaneous Delivery and Pickup

VSM : Vector Space Model

Introduction générale

Sommaire

1 Contexte et problématique

Avec l'avancée technologique spectaculaire de l'Intelligence Artificielle (IA) dans plusieurs domaines, l'exploitation ou l'analyse de données volumineuses est devenue une tâche essentielle. L'évolution du Web a été marquée par une forte croissance des services publiés qui s'est accompagnée d'une explosion considérable du nombre d'usagers dont leurs contraintes et leurs préférences sont diverses et variées. L'analyse de cette quantité d'informations devient importante afin d'extraire de l'information pertinente. Ainsi, trouver le service qui satisfait mieux les besoins et les préférences de l'utilisateur est un enjeu important.

Plusieurs techniques de l'IA ont été développées pour faciliter la recherche et l'extraction d'informations pertinentes. Les techniques d'optimisation et de recommandation ont connu un grand intérêt en raison de leur bonne performance sur les grands volumes de données et leur capacité d'analyse et d'extraction d'informations pertinentes afin de l'adapter aux attentes des utilisateurs.

Les techniques d'optimisation consistent en un ensemble de méthodes de résolution de problèmes d'optimisation combinatoire (POC) qui se subdivisent en trois groupes : les méthodes exactes, les méthodes heuristiques et les méthodes hybrides. Les algorithmes exacts garantissent de trouver la solution optimale et de prouver son optimalité pour toutes les instances d'un problème. Par contre, le temps nécessaire pour trouver la solution optimale d'un POC augmente en général fortement avec la taille du problème. De plus, pour certains problèmes, la consommation de mémoire de ces algorithmes peut être très grande et peut parfois entraîner l'arrêt prématuré de l'application informatique. Les méta-heuristiques gagnent maintenant en popularité, car les meilleurs résultats trouvés pour plusieurs POC ont étés obtenus avec des algorithmes hybrides qui impliquent une combinaison de plusieurs méta-heuristiques ou une combinaison d'une méthode exacte et d'une méta-heuristique. L'utilisation des méthodes heuristiques permet d'obtenir des solutions de bonnes qualités en un temps de résolution raisonnable.

Les SR sont capables de fournir des recommandations basées sur les préfé-

rences et les besoins des utilisateurs. Ils se sont avérés fournir des résultats très satisfaisants et fournir un excellent soutien aux utilisateurs. Les SR sont rapidement devenus très populaires dans divers domaines en raison de leurs hautes performances. La prise en compte de toutes les informations relatives au demandeur de services peut avoir un impact crucial sur l'appréciation de l'utilisateur des résultats retournés.
Notre propos consiste à valoriser cet impact dans ce cadre d'optimisation et de recommandation de services. Les problématiques abordées dans le cadre de notre thèse sont caractérisées par deux points :

1. *Problème d'optimisation combinatoire (le plus court chemin) :* Les problèmes d'optimisation combinatoire sont présents dans plusieurs domaines d'applications. Notre problématique consiste à chercher un chemin élémentaire de coût minimal entre deux sommets d'un graphe. Le graphe consiste à la position actuelle (p) du demandeur de services et un ensemble de lieux touristiques (t) (destination souhaitée). Ce problème appartient à la classe NP-complet car nous avons implémenté un graphe orienté de 10 000 noeuds et un autre non orienté de 5000 noeuds. Le problème devient plus compliqué si le nombre de sommets est très important. Dans ce cas, on fait appel aux méta heuristiques afin de résoudre ce genre de problèmes. Une solution s à ce problème est un chemin élémentaire de p à t. L'ensemble S est l'ensemble de tous les chemins élémentaires possibles de p à t. Cet ensemble est bien fini, il sera utilisé dans notre deuxième contribution afin d'améliorer le processus de recommandation.

2. *Recommandation du meilleur item :* La croissance explosive des services sur Internet et la diversité des préférences des utilisateurs ont conduit à une difficulté à sélectionner les meilleurs services qui répondent aux besoins de l'utilisateur en termes de préférences et du contexte. De plus, les demandeurs de services sont souvent confrontés à de nombreux items concurrents qui offrent des fonctionnalités *similaires* mais ils sont associés à des contraintes contextuelles *différentes*. Par conséquent, les utilisateurs doivent sélectionner les meilleurs items de la liste avec les fonctionnalités requises et la plus haute qualité souhaitée. Une recommandation personnalisée doit collecter des données sur les utilisateurs et les items

disponibles pour prédire les meilleurs items. Des systèmes de recommandation personnalisés ont été mis en place dans de nombreuses applications réelles où ils ont prouvé avoir du succès dans le cinéma, la musique, les réseaux sociaux, la santé, l'actualité personnalisée, etc., mais semblent limités dans le tourisme. Dans le domaine du tourisme, la recommandation peut être faite dans diverses situations telles que la recommandation d'itinéraires intelligents (notre première contribution [M. Y. Hayi, 2022]), la recommandation des points d'intérêt (POI) (notre deuxième contribution [Z. Chouiref, 2022]), etc.

Les SR de services touristiques existants peuvent recommander des services non adaptés au contexte du demandeur de services en temps réel, par exemple, un visiteur peut avoir dans la liste recommandée du *tourisme de montagne* en un *très mauvais climat* au lieu de lui recommander de visiter un *musé* dans ces *mauvaises conditions météorologiques*. Notre problématique est comment tenir compte à la fois de quelques aspects importants liés au contexte de la destination visitée en temps réel (sa localisation et sa situation météorologique) et des informations du profil et des préférences relatives à l'utilisateur afin d'améliorer le processus de recommandation et augmenter sa précision prédictive dans le domaine du tourisme ?

2 Motivations et Contributions

Motivé par les problématiques mentionnées ci-dessus, en contraste aux approches existantes qui se concentrent uniquement sur les informations contextuelles explicites, dans notre travail, nous proposons une approche efficace et intelligente de recommandation des points d'intérêts touristiques, qui offre aux utilisateurs un moyen adéquat pour tenir compte de leurs préférences *explicites/implicites* et améliorer la recommandation de service en s'appuyant sur son contexte extraits en *temps réel*.

Le domaine d'application retenu dans cette thèse se situe à l'intersection du paradigme d'*optimisation combinatoire* et des *systèmes de recommandation* utilisant les *techniques de machine learning*. Les algorithmes d'optimisation s'utilisent

en de nombreux problèmes. Un algorithme d'optimisation consiste à une procédure mathématique qui permet d'obtenir les minimums (ou maximums) d'une fonction objective. Le but de l'optimisation combinatoire dans notre travail est de sélectionner les chemins optimaux - parmi ceux existants - qui répondent rapidement à la demande de l'utilisateur par rapport à sa position, en se basant sur l'hybridation des algorithmes exacts et des méthodes méta-heuristiques.

Le système de recommandation proposé est construit suivant un modèle sensible à trois dimensions : le profil, les préférences et le contexte. Le profil de l'utilisateur contient ses informations personnelles comme l'âge, le genre et d'autres informations. Les préférences changent d'un utilisateur à un autre, pour cela le système doit respecter les préférences et recommander des items d'après le besoin, le choix et le goût des utilisateurs. Le contexte peut contenir la position GPS de l'utilisateur. Notre deuxième intérêt de recherche porte sur l'exploitation du contexte météorologique du POI et des informations relatives aux préférences de l'utilisateur dans la phase de recommandation. Un bon mécanisme de recommandation de POI ne devrait pas permettre d'exploiter que les préférences que l'utilisateur définit dans sa requête, mais aussi d'en déduire des préférences supplémentaires à partir de sa situation. Enrichir également la requête de l'utilisateur par des contraintes contextuelles comme la situation météorologique et la position GPS de la destination souhaitée extraites en temps réel, permet d'améliorer fortement la qualité des résultats.

La sensibilité au contexte dans notre approche consiste en deux points : (i) La localisation : qui consiste à trouver un ou plusieurs courts chemins ç-à-d calculer la meilleure distance entre la source (position de l'utilisateur) et la destination (lieu touristique souhaité). L'item le plus proche a plus de chance d'être sélectionné qu'un item plus loin. (ii) La météo : qui consiste à la récupération automatique de la situation météorologique des différents lieux touristiques, afin de proposer les meilleurs selon les conditions de l'humidité et du climat du point d'intérêt.

Une proposition d'approches intelligentes et cruciales basées sur les techniques de machine learning seront d'utilité importante. Dans ce qui suit, nous allons

présenté brièvement l'approche d'optimisation [M. Y. Hayi, 2022] et l'approche de recommandation [Z. Chouiref, 2022] proposées.

Dans [M. Y. Hayi, 2022], nous avons développé notre propre algorithme d'optimisation IOGA (Improved Optimization Genetic Algorithm) qui permet de retourner un ou plusieurs courts chemins entre une source et une destination.
Pour valider l'algorithme IOGA, nous avons créé deux bases de données, une orientée et l'autre non orientée, sur lesquelles nous avons appliqué quatre algorithmes, à savoir : IOGA, Dijkstra, algorithme génétique et A*. Les résultats obtenus montrent l'efficacité d'IOGA. Ils sont acceptables surtout dans les graphes de grande taille par rapport aux (i) plus court chemin, (ii) nombre des chemins retournés, (iii) temps d'exécution et (iiii) complexité.

La recommandation fait aussi une partie importante dans notre projet, et IOGA est inclus dans la recommandation, puisque nous avons calculé les plus courts chemins par ce dernier dans la phase de recommandation. Un bon système de recommandation doit sélectionner les items qui peuvent satisfaire les clients et ignorent les autres. Les clients ne peuvent pas être satisfait s'ils visitent des points d'intérêts qui ne les intéressent pas. Inversement, ils peuvent manquer des items ou des points d'intérêts qui auraient pu les intéresser.

Dans [Z. Chouiref, 2022], un système de recommandation appelé H-RN (Hybrid Random Forest and Naïve Bayes) a été proposé. Il est appliqué sur quatre jeux de données avec quatre autres algorithmes de machine learning à savoir : Random Forest, Naive Bayes, K-NN et Neural Network. Ainsi, l'ajout de la partie météorologique à améliorer la qualité des résultats.
Plusieurs expérimentations sont appliquées pour vérifier et valider l'efficacité de notre proposition. Les résultats obtenus sont satisfaisants par rapport à la qualité de recommandation, un taux d'erreur faible, une complexité et un temps de réponse acceptables.

Puisque nous sommes dans l'ère de la vitesse, il est nécessaire de penser à la rapidité du fonctionnement de notre système, pour cela nous avons étudié la complexité des deux algorithmes proposés et aussi nous avons calculé le temps

d'exécution sur la majorité de nos expérimentations afin de les comparer avec d'autre algorithmes.
Les différentes parties et scripts de notre projet sont implémentés avec le langage python à savoir le traitement des données, la création des jeux de données d'IOGA, le calcul des mesures d'évaluations, la création des graphes de comparaisons, etc.

Pour résumer, l'objectif principal de l'approche suggérée consiste à utiliser des techniques et des algorithmes de machine learning qui peuvent prédire et proposer des services touristiques pertinents (k-items) aux utilisateurs selon leurs intérêts, leurs besoins, leurs goûts et leurs contextes.
Dans cette recherche, nous décrivons comment les techniques d'apprentissage peuvent fournir automatiquement des recommandations personnalisées aux demandeurs en tenant compte à la fois des informations sur leurs préférences et de leur contexte implicite/explicite et les contraintes contextuelles en temps réel des points d'intérêt. Nous construisons un algorithme H-RN efficace et intelligent qui hybride à la fois l'apprentissage automatique le plus connu algorithmes, à savoir Random Forest et Naïve Bayes, avec un filtrage collaboratif (technique basée sur le modèle et basée sur la mémoire). Différentes expériences de notre démarche dans le cadre d'un dispositif de recommandation dans le domaine touristique s'effectuent sur les quatre grands ensembles de données du monde réel.
Les systèmes de recommandation peuvent utiliser H-RN pour améliorer prédiction de recommandation et réduire l'espace de recherche des services touristiques. De plus, le résultats du rappel, de la précision, de l'exactitude, de la mesure F et du taux moyen, ainsi qu'un ensemble de des tests statistiques (One-way ANOVA, Diversity) et des mesures d'erreur (RMSE, MAE) ont été discuté pour montrer l'amélioration de la précision de prédiction de notre algorithme par rapport aux approches de base dans divers contextes.

3 Organisation du manuscrit

Notre thèse s'articule autour de deux parties et de cinq chapitres. La première partie de l'état de l'art est composée de trois chapitres. Dans le premier chapitre, nous présentons un aperçu général sur l'optimisation, le concept du plus court chemin, la théorie des graphes et nous introduisons ensuite les graphes orientés et non orientés, ainsi que nous présentons les différents algorithmes connus d'optimisation.

Le deuxième chapitre présente l'état de l'art des systèmes de recommandation. Nous illustrons les différentes mesures d'évaluation des SR ainsi que nous présenterons dans ce chapitre plusieurs algorithmes connus dans ce domaine. Sans oublier les différentes approches de recommandation.

Le troisième chapitre contient les travaux connexes récents qui nous permettent de faire une comparaison générale entre les différents travaux, d'après les bases de données utilisées ou les techniques et les algorithmes et même pour voir les résultats obtenus avec leurs analyses.

La deuxième partie représente les contributions. Elle contient deux chapitres, le premier chapitre représente la partie d'optimisation, où nous avons développé l'algorithme IOGA et nous l'avons appliqué sur deux data sets que nous avons créé. Nous avons aussi présenté les résultats obtenus et une étude comparative avec d'autres algorithmes.

Le deuxième chapitre décrit notre système de recommandation proposé avec l'explication détaillée de l'algorithme H-RN et la sensibilité au contexte par rapport à la situation météorologique, les quatre bases de données utilisées avec leurs sources, ainsi que nos différents résultats obtenus.

4 Liste des publications

A) REVUE :

1. Mohamed Yassine HAYI, Zahira CHOUIREF, Hamouma MOUMEN. To-

wards Intelligent Road Traffic Management Over a Weighted Large Graphs Hybrid MetaHeuristic-Based Approach. *Journal of Cases on Information Technology*. Volume 24, Issue 3, DOI : 10.4018/JCIT.20220701.oa4, 2022 [M. Y. Hayi, 2022].

2. Zahira CHOUIREF, Mohamed Yassine HAYI. Toward Preference and Context-Aware Hybrid Tourist Recommender System Based on Machine Learning Techniques. *Revue d'Intelligence Artificielle*. Vol. 36, No. 2, April, 2022, pp. 195-208 [Z. Chouiref, 2022].

B) CONFERENCE INTERNATIONALE :

1. Mohamed Yassine HAYI, Zahira CHOUIREF. An Improved optimization Algorithm to Find Multiple Shortest Paths over Large Graph. *IEEE/ Second International Conference on Embedded Distributed Systems (EDiS) 2020*, pp. 178-182 [M. Y. Hayi, 2020].

2. Mohamed Yassine HAYI, Zahira CHOUIREF. An Efficient Hybrid Metaheuristic Approach for Solving the k-Shortest Paths Problem over Weighted Large Graphs. *Springer/ 4th International Conference on Artificial Intelligence in Renewable Energetic Systems (IC-AIRES)* December 22-24, 2020 [M. Y. Hayi, 2021].

Première partie

Etat de l'art

Chapitre 1

Optimisation

Sommaire

1 Introduction

Avec la croissance massive de l'information, l'optimisation dans le Big Data est devenue l'un des principaux défis dans la recherche et l'une des sciences les plus utilisées dans divers domaines. De plus, des réseaux complexes tels que les réseaux de transport, de communication, de distribution et Internet, etc., ont de plus en plus attiré l'attention de diverses applications de la science et de l'ingénierie notamment le domaine d'optimisation.

Un problème d'optimisation combinatoire est un problème algorithmique dans lequel l'objectif est de maximiser ou minimiser la solution parmi un ensemble de solutions également valides mais potentiellement moins bonnes, sachant que chaque problème a ses propres critères, objectifs, méthodes et algorithmes. Ce type de problème est caractérisé par une fonction d'évaluation de qualité ou un degré de chaque solution.

Face à de nouveaux problèmes complexes d'optimisation, il est primordial d'utiliser des techniques méta-heuristiques [M. Nasr, 2020]. La recherche en méta-heuristique est bien connue dans le domaine de l'intelligence artificielle et a également été très active au cours des dernières décennies. Combiner une méta-heuristique avec d'autres techniques d'optimisation, peut fournir une flexibilité avancée et un comportement plus efficace face à des problèmes à grande échelle et réels [M. Nasr, 2020].

De nombreux algorithmes d'optimisation ont été proposé dans la littérature, et diverses applications telles que les applications de transport nécessitent l'utilisation d'une méta-heuristique du plus court chemin plutôt qu'un des algorithmes standards existants. Le problème du plus courts chemins est un problème d'optimisation combinatoire qui a été largement étudié dans la littérature. Nous pouvons citer quelques exemples d'applications de manière non exhaustive :

— réseaux de transports routier, d'eau, d'électricité, etc.

— réseaux informatiques où les sommets représentent les ordinateurs et les arêtes les connexions physiques ;

— graphe du web où les sommets représentent les pages web et chaque arc cor-

respond a un hyperliens d'une page vers une autre ;
— réseau de transports de données (téléphonie, wifi, réseaux informatique, etc.) ;
— et beaucoup d'autres encore, etc.
La théorie des graphes est l'un des paradigmes utilisés pour représenter ce type de problème. Le graphe est un réseau complexe important qui permet de décrire la relation entre un ensemble de nœuds inter-connectés (entités). La taille du graphe étudiée dans notre travail est très importante. De ce fait, les algorithmes classiques ne sont pas appropriés pour les graphes de plus grande taille. L'algorithme heuristique qui trouve le chemin le plus court doit être optimal et significatif en terme de rapidité et les chemins efficaces.

Dans ce chapitre, nous présentons tout d'abord l'optimisation et son historique ainsi que ses buts. Ensuite, nous détaillons la notion de théorie des graphes et sa relation avec l'optimisation à savoir les graphes orientés et non-orientés ainsi que le problème du plus court chemin. Enfin, nous terminons par une présentation des différents algorithmes les plus connus dans l'optimisation comme les algorithmes génétiques, Dijkstra, A*, Floyd-Warshall et les algorithmes heuristiques.

2 Définition d'optimisation

L'optimisation est une technique mathématique utilisée pour trouver un minimum ou un maximum global permettant d'optimiser une solution à un problème donné. Optimiser, c'est-à-dire chercher rapidement la meilleure solution avec un minimum de coûts [Ketfi, 2016]. Par exemple, d'un point de vue commercial, un profit maximal est souhaité à moindre investissement. L'optimisation est caractérisée par trois étapes générales [A. J. Kulkarni, 2017] : modélisation, analyse, et résolution des problèmes.

2.1 Modélisation

La première étape de résolution d'un problème est la modélisation des données, par conséquent, un bon modèle affecte la qualité de la solution. En optimisation,

généralement on remplace les informations à optimiser par un graphe ou une matrice d'adjacence ou matrice d'incidence, cela permet d'appliquer facilement les différents algorithmes d'optimisation.

2.2 Analyse

L'analyse des données (aussi appelée analyse exploratoire des données) est une partie très importante dans un processus d'optimisation. La phase d'analyse a pour objectif de supprimer des mots vides et des données non importante du data-set, afin de bien l'organiser et le faire passer à la prochaine étape. Cette partie nécessite l'utilisation de quelques techniques et quelques scripts de programmation, c'est une partie compliquée qui nécessite beaucoup de temps surtout dans le cas des bases de données volumineuses.

2.3 Résolution des problèmes

C'est la pulpe de l'optimisation, il existe deux méthodes de résolution de problèmes : analytique et numérique, ainsi qu'il existe des dizaines d'algorithmes. Cette dernière partie nécessite des processeurs rapides pour gagner du temps, par exemple, le problème du plus court chemin dans un graphe volumineux nécessite environ trois heures pour trouver un meilleur chemin dans une base de données de cent mille nœuds si nous utilisons un simple processeur, contrairement à un processeur rapide et utile, qui nécessite seulement quelques minutes.

3 Historique d'optimisation

Vers 300 ans avant notre ère, le mathématicien de la Grèce antique Euclide d'Alexandrie a formulé les premiers problèmes d'optimisation. Il a proposé une méthode permettant de trouver le plus court chemin entre deux point A et C en passant par un point de la droite, ce chemin est obtenu lorsque l'angle d'incidence est égal à l'angle réfléchi. La figure (1.1) suivante représente une explication de cette formulation [Pape, 1974].

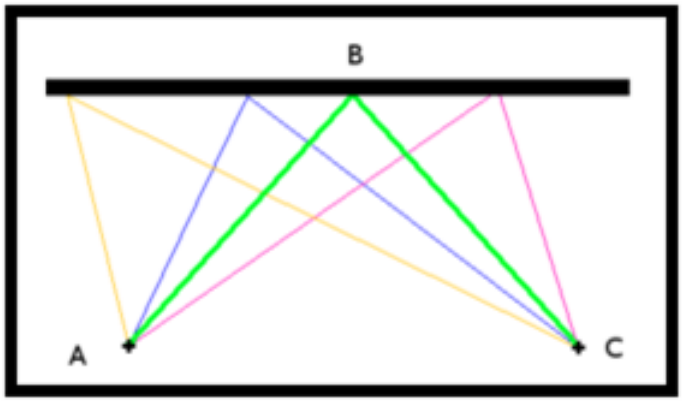


FIGURE 1.1 – Explication du premier problème d'optimisation

En deuxième siècle avant notre ère, le mathématicien Euclide venait de formuler le premier problème d'optimisation dans son ouvrage historique Éléments. Des années plus tard, Héron d'Alexandrie (ingénieur, mécanicien et mathématicien grec) établissait ses premiers énoncés sur le principe du plus court chemin [S. Maguire, 2006].

Au XVIIe siècle, Newton prouvait le besoin d'établir des techniques d'optimisation pour l'amélioration du calcul différentiel, et après plusieurs tentatives, il développa une méthode itérative afin d'augmenter la probabilité d'avoir la meilleure valeur approximative du calcul différentiel. Cette méthode itérative a permis d'améliorer les résultats et les fonctionnements [Bellman, 1958].

Depuis, l'optimisation a connu un développement fulgurant et a commencé à être utilisée dans plusieurs domaines [Polyak, 2007]. L'optimisation est l'outil privilégié pour identifier les conditions d'opérations les plus souhaitables ou le programme des opérations le plus rentable menant à la meilleure performance, et ce dans le respect des contraintes. Des exemples de problèmes fréquents sont : mélange de produits à coût minimal, confection d'horaire de production, maximisation de rendement, minimisation des pertes et augmentation de débit dans un réseau.
La science de l'optimisation est généralement comprise et pratiquée par les mathématiciens, alors que les problèmes d'optimisation sont issus des activités des ingénieurs.

4 Théorie des graphes et Optimisation

La théorie des graphes est une discipline mathématique et informatique. Elle s'occupe de l'étude des graphes et permet de travailler sur les relations entre les données. Le mathématicien suisse Leonhard Euler est considéré à l'origine de la théorie des graphes car ayant créé le premier graphe pour résoudre le problème des sept ponts de Königsberg en 1735, et publié un article en la matière en 1741. Le problème des sept ponts de Königsberg est très simple, la petite ville russe est construite autour de deux îles reliées par un pont, et il existe six autres ponts reliant les deux rives [H. Sachs, 1988].

La figure suivante représente l'image, dessin et graphe illustrant ce problème connu.

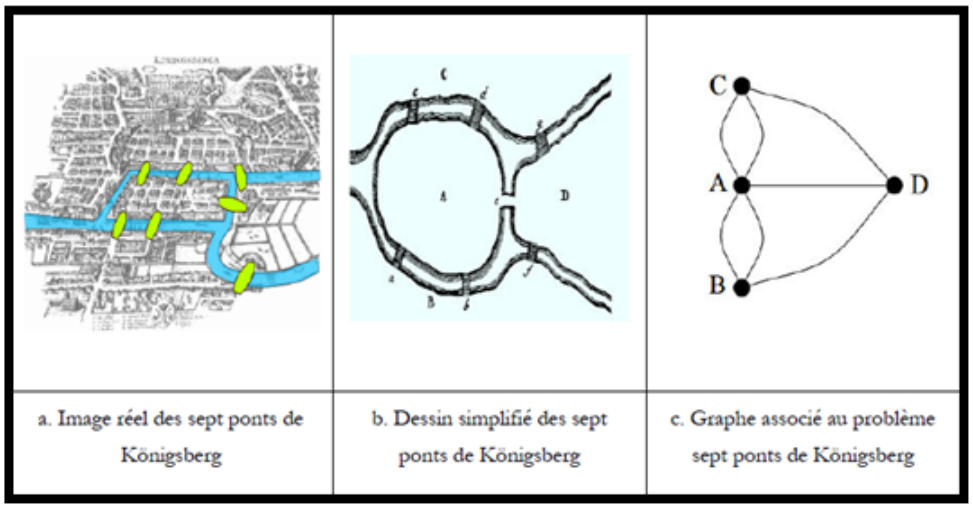


FIGURE 1.2 – Problème des sept ponts de Königsberg

Le problème consiste à déterminer s'il existe ou non un chemin à partir d'un point de départ au choix, de passer une et une seule fois par chaque pont, et de revenir à son point de départ [N. L. Biggs, 1998] [J. L. Gross, 2013].

La théorie des graphes étudie alors les nombreuses propriétés de ces représentations. Il s'agit de l'existence de chemins les plus courts, les chemins les moins coûteux, le nombre d'intersections dans le plan, les problèmes de coloriage, les cycles particuliers, etc.

4.1 Graphes

Le graphe est défini comme une collection d'éléments qui sont mis en relation entre eux. Leur représentation géométrique se fait à travers des modèles constitués par des points (appelés encore sommets ou nœuds) reliés par des arrêtes (également appelés liens ou lignes de courbes). Les arêtes peuvent être non symétriques et sont alors considérées comme des flèches ou des arcs. Quand on choisit de les orienter et/ou leur attribuer un poids, les graphes sont dits orientés ou pondérés. Il existe deux types de graphes :

4.1.1 Graphe orienté

Un graphe orienté est formé par un nombre donné de nœuds reliés entre eux par des arcs où chaque arc a une direction, donc nous pouvons trouver un arc du nœud A vers un nœud B où un autre arc reliant un nœud B au nœud A. Chaque arc peut avoir un poids, Si ce dernier n'est pas indiqué sur la flèche, alors il est égal à un [L. Georgiadis, 2014].

L'exemple suivant contient une représentation mathématique d'un graphe orienté (figure 1.3).

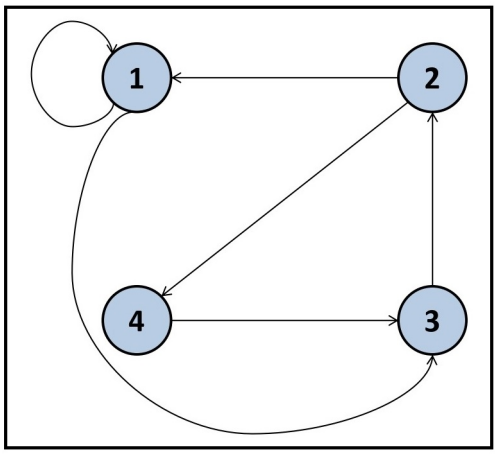


FIGURE 1.3 – Graphe orienté explicatif de la formule mathématique

Sachant que :

G = (S,A) où

- S = {1,2,3,4},

- A = {(1,1), (1,3), (2,1), (2,4), (3,2), (4,3)}.

La figure (1.4) représente un graphe orienté où chaque arc a son poids.

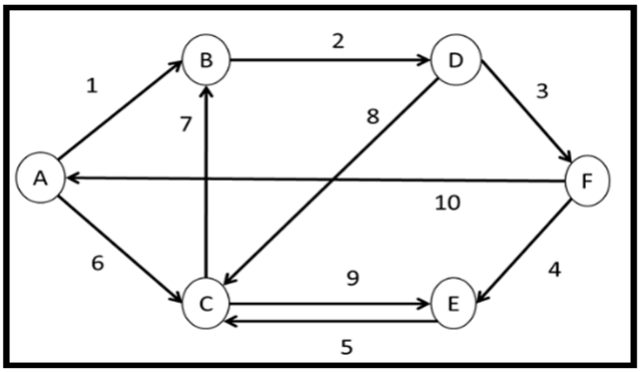


FIGURE 1.4 – Exemple d'un graphe orienté

4.1.2 Graphe non-orienté

Un graphe non-orienté représente une liste de nœuds reliés entre eux par des arcs mais un seul arc représente la relation réciproque entre deux nœuds, donc nous ne pouvons pas avoir deux arcs entre deux nœuds [C. G. Rodriguez, 2012] [A. K. Jeewajee, 2020]. L'exemple suivant contient une représentation mathématique d'un graphe non-orienté (figure 1.5).

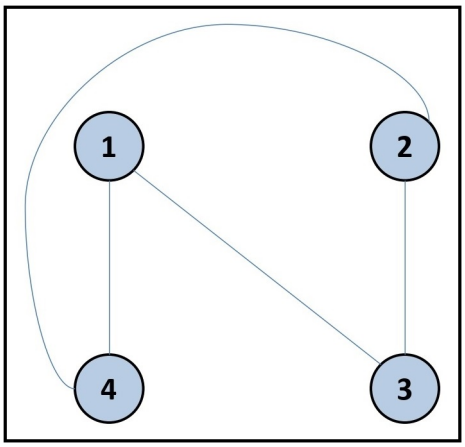


FIGURE 1.5 – Graphe non-orienté explicatif de la formule mathématique

Sachant que :

G = (S,A) où

- S = {1,2,3,4},

- A = {(1,3), (1,4), (2,3), (2,4)}.

La figure (1.6) représente le graphe des stations de métro à la ville de Paris. D'après ce graphe, nous pouvons trouver le meilleur chemin entre deux stations ou le meilleur chemin pour visiter toutes les stations.

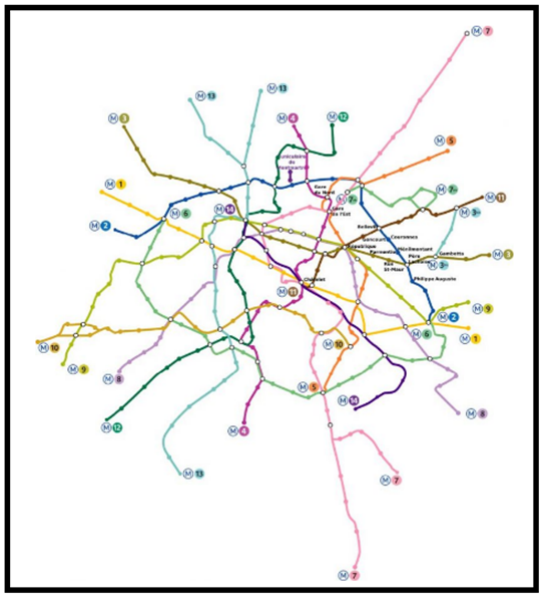


FIGURE 1.6 – Exemple d'un graphe non-orienté

D'après la figure, nous nous apercevons que tout le monde forme un réseau connecté, soit par les routes (routes automobiles, route aérienne, routes maritimes mondiales) voire par la connectivité que nous offre la toile d'Internet. La figure (1.7) représente un graphe de la connectivité dans les différents points de la terre.

4.2 Concept du plus court chemin

Le problème des K-chemins les plus courts est la base de nombreux problèmes d'optimisation combinatoire. C'est l'un des problèmes les plus connus dans le domaine de l'informatique, et largement utilisé dans divers domaines, tels que :

FIGURE 1.7 – Graphe de la connectivité entre les différents points de la terre

le routage de réseaux informatiques, robot Pathfinder, itinéraire de navigation, conception de jeux, etc [I. Maatouk, 2017].

Depuis bien des années, il existe des dizaines d'algorithmes pour résoudre ce problème. Le meilleur exemple est le fonctionnement du GPS (Global Positioning System) pour proposer un chemin plus court et éviter ainsi les points noirs de la circulation. Tout simplement, le système doit disposer d'une liste des routes disponibles et y chercher le meilleur chemin ou le plus court chemin qui satisfait rapidement le besoin de l'utilisateur [M. Changxi, 2018]. Dans le problème du plus court chemin, nous cherchons toujours à réduire la distance entre la source et la destination [Guillot, 2020] [Manseur, 2017].

Pour formuler un modèle mathématique pour trouver les K chemins les plus courts entre le nœud d'origine et le nœud de destination, nous supposons que G = (N, E, C), un graphe orienté pondéré avec un ensemble de sommets (nœuds) (N) et un ensemble d'arêtes dirigées (E) et les valeurs de coût d'arête (C), telles que :

- c(s, t) : coût de l'arrête dirigée d'un nœud source « s » vers un nœud cible « t » (s et t sont à la fois dans N) sachant que les coûts soient non négatifs.
- Les chemins renvoyés sont les plus courts possibles.

Le problème des K chemins les plus courts est une généralisation naturelle et longuement étudiée, dans lequel on cherche non pas un seul chemin mais plusieurs chemins par ordre croissant de longueur. Le problème de la détermination des plus courts chemins K s'est avérée plus difficile. Pour trouver le chemin le plus court, nous pouvons utiliser les algorithmes du plus court chemin qui peuvent être classés comme suit *(i) All Even Shortest Path Algorithms (APSP)* or *(ii) Single Source Shortest Path Algorithms (SSSP)*. Dans notre travail, nous nous intéressons aux algorithmes *SSSP* qui trouvent les k chemins avec le coût minimal à partir d'un seul sommet d'origine à tous les sommets de destination du graphe. L'algorithme de Dijkstra [Dijkstra, 1959] ; Bellman-Ford [Bellman, 1958] [Ford, 1956] et l'algorithme de D'Esopo-Pape [Pape, 1974] en sont quelques exemples des algorithmes *SSSP* bien établis. L'algorithme de Dijkstra requiert le calcul des chemins les plus courts entre les nœuds d'origine et tous les nœuds les plus proches de la matrice Origine-Destination. L'algorithme de Dijkstra peut résoudre ce problème et il peut être étendu pour trouver plus d'un chemin. L'algorithme génétique a d'excellentes performances dans la recherche d'espaces avec de grandes solutions, et peut faire des compromis entre efficacité et précision. Dans notre travail, nous avons essayé de résoudre ce problème en appliquant l'algorithme de Dijkstra et l'algorithme génétique (détails de l'approche est présentée dans le **chapitre 4** de la partie *Contributions*).

5 Quelques Algorithmes d'optimisation

Ils existent plusieurs algorithmes exacts, hybrides, heuristiques ou même métaheuristiques utilisés dans le domaine de l'optimisation. Les différents algorithmes existants ont les mêmes objectifs, à savoir optimiser rapidement les résultats d'un problème donné. Le concept de ces algorithmes change selon le domaine de l'utilisation ainsi que la taille des choix qu'offre un problème donné. Chaque algorithme a ses propres méthodes et techniques et cette diversité permet de créer une concurrence de qualité entre les algorithmes. Nous présentons ici quelques algorithmes connus et très utilisés dans le domaine d'optimisation :

5.1 Algorithme de Dijkstra

L'algorithme de Dijkstra est l'algorithme le plus populaire dans le problème du plus court chemin. Il est souvent utilisé pour orienter des graphes pondérés par des réels positifs. Très ancien mais très fonctionnel, il a été mis en place en 1959 par l'informaticien néerlandais Edsger Dijkstra. Cet algorithme tente toujours à trouver le chemin le plus court d'un nœud à tous les autres nœuds. Il prend un nœud initial, et cherche le chemin le plus court vers tous les autres nœuds. Dans une seconde itération, en fonction des nœuds visités par le premier nœud, il choisit le chemin le plus court du nœud précédent et répète la première étape, et ainsi de suite jusqu'à ce qu'il active tous les nœuds du graphe [Singh, 2018] [Grabener, 2011]. L'algorithme 1 représente l'algorithme de Dijkstra en détail.

Algorithm 1 Algorithme de Dijkstra

```
Input Graphe, Source, Destination
Output Meilleure solution
Begin
Create vertex set Q
For each vertex set Q
    Dist[v] ← INFINITY
    Prev[v] ← UNDEFINED
    Add v to Q
End for
Dist[source] ← Q
While Q is not empty do
    U ← vertex in Q with min dist[u]
    Remove u from Q
    For each  neighbor v of u
        Alt ← dist[u] + lenght(u,v)
      If alt ≤ dist[v] Then
          Dist[v] ← alt
          prev[v] ← u
      End If
    End For
End While
Return(dist[ ],prev[ ])
End
```

5.2 Algorithme Floyd-Warshall

Floyd-Warshall ou aussi Roy-Floyd-Warshall est un algorithme qui détermine les distances des plus courts chemins entre n'importe quelles paires de sommets dans un graphe orienté et pondéré. il a été décrit par Bernard Roy en 1959 [Bernard, 1959]. Ce dernier est l'un des algorithmes connus dans le domaine de l'optimisation, son fonctionnement est facile à comprendre. Nous prenons tous les arcs du graphe, chaque arc représente une relation entre deux nœuds de type k=0, et nous ajoutons de nouvelles relations par la combinaison des arcs entre eux, et ainsi de suite [L. Xing, 2019]. La figure (1.8) explique mieux le fonctionnement de l'algorithme Floyd-Warshall.

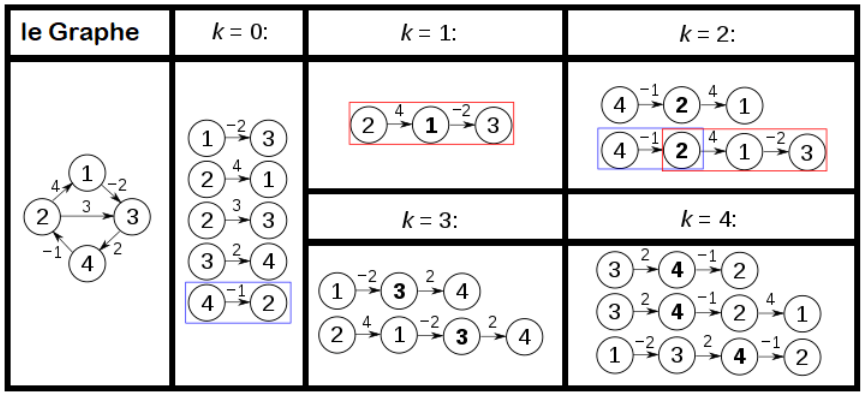


FIGURE 1.8 – Exemple sur l'algorithme de Floyd-Warshall

5.3 Algorithmes Méta Heuristiques

Une méta-heuristique est une méthode générique pour la résolution des problèmes combinatoires. La résolution de ces problèmes nécessite l'examen d'un très grand nombre (exponentiel) de combinaisons. Plusieurs chercheurs ont déjà été confrontés à ce phénomène d'explosion combinatoire qui transforme un problème apparemment très simple en un véritable casse-tête dès lors que nous augmentons la taille du problème à résoudre. C'est le cas par exemple quand on cherche à concevoir un emploi du temps. S'il y a peu de cours à planifier, le nombre de combinaisons à explorer est faible et le problème est très rapidement résolu. Cependant, l'ajout de quelques cours seulement peut augmenter considérable-

ment le nombre de combinaisons à explorer de sorte que le temps de résolution devient excessivement long [H. Kashif, 2019]. Le but d'une méta-heuristique, est de réussir à trouver un optimum global. Pour cela, l'idée est à la fois de parcourir l'espace de recherche, et d'explorer les zones qui paraissent prometteuses, par contre les heuristiques sont des méthodes de résolution purement algorithmiques qui permettent d'obtenir des solutions à n'importe quel problème décisionnel rapidement.

Les méta-heuristiques contiennent plusieurs algorithmes, nous présentons ci-après les plus connus :

5.3.1 Algorithme Génétique

Un algorithme génétique est un algorithme de recherche et d'optimisation aléatoire et évolutif qui imite l'évolution biologique naturelle. Il a été appliqué pour résoudre divers problèmes dans différents domaines d'activité humaine et utilisé pour obtenir une solution optimale dans un délai raisonnable. Il utilise une sélection naturelle (aléatoire) pour avoir une population. Ce dernier évoluera après chaque itération, et lorsque l'algorithme atteint une condition d'arrêt, il retournera la meilleure solution qui existe dans la population comme solution finale au problème [Hamed, 2010], [Hosseinabadi, 2018]. La figure (1.9) représente l'organigramme de l'algorithme génétique.

5.3.2 Algorithme A*

L'algorithme A* est considéré comme l'un des heuristiques les plus populaires en raison de sa complétude, de son optimalité et de son efficacité optimale [H. Yusnita, 2020] [Kacem, 2012]. Il est très utile dans le domaine d'optimisation, et se fraye un chemin d'un point à un autres et se dirige vers la destination. À chaque itération, il va essayer de se rapprocher de la destination. On privilégiera donc les possibilités qui sont directement plus proches de la destination, en laissant de côté toutes les autres [A. K. Guruji, 2016]. L'algorithme A* qui est connu aussi par le nom A-star est compliqué, car il est formé par plusieurs étapes :

a/ Lire le graphe.

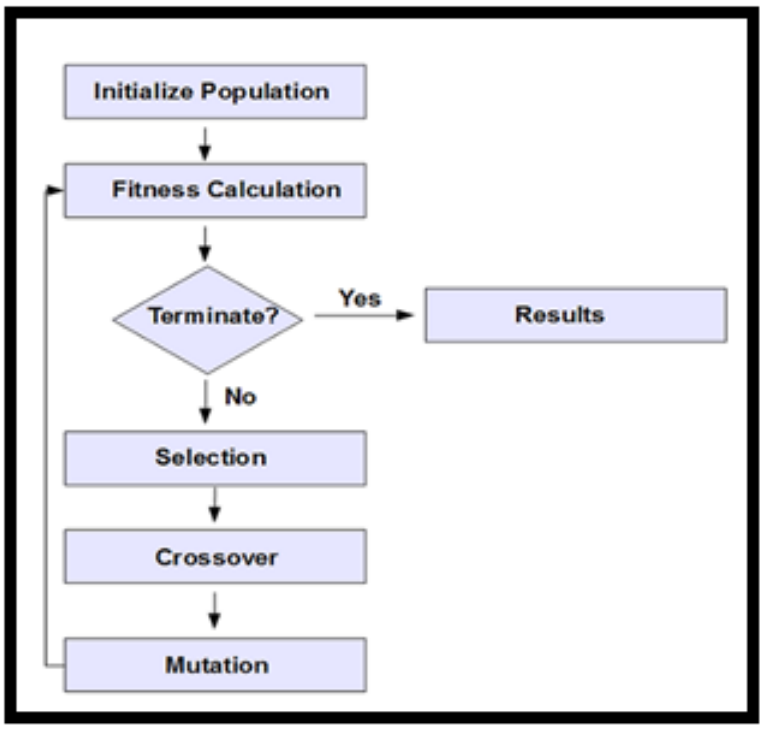


FIGURE 1.9 – Organigramme de l'algorithme génétique

b/ Récupérer les chemins existants.

c/ Vérifier si un emplacement valide existe.

d/ Ignorer la position du nœud si elle est invalide.

e/ Ajouter à la liste des nœuds enfants valides pour le parent sélectionné.

f/ Obtenir les meilleures résultats.

La figure (1.10) illustre en détail la partie de la génération des enfants de toutes les cases adjacentes de l'algorithme A* dans un script python.

```
# Generate children from all adjacent squares
children = []

for new_position in move:

    # Get node position
    node_position = (current_node.position[0] + new_position[0],
                     current_node.position[1] + new_position[1])

    # Make sure within range (check if within maze boundary)
    if (node_position[0] > (no_rows - 1) or
        node_position[0] < 0 or
        node_position[1] > (no_columns -1) or
        node_position[1] < 0):
        continue

    # Make sure walkable terrain
    if maze[node_position[0]][node_position[1]] != 0:
        continue

    # Create new node
    new_node = Node(current_node, node_position)

    # Append
    children.append(new_node)
```

FIGURE 1.10 – Script python de la génération des enfants dans A*

5.3.3 Algorithme Gloutons

Un algorithme glouton (en anglais greedy algorithms) est un algorithme qui effectue le meilleur choix possible à chaque instant. Il fonctionne donc étape

par étape sans avoir à revenir en arrière ou à prédire l'étape suivante. L'algorithme glouton cherche une solution optimale locale à chaque étape, dans le but de trouver la meilleure solution optimale. Les algorithmes gloutons sont parfois appelés algorithmes gourmands. Ils sont très utilisés dans les problèmes de voyageur de commerce, l'arbre couvrant minimal, coloration de graphe et même dans le problème du rendu de monnaie. Par exemple, si nous souhaitons résoudre le problème du rendu de monnaies par l'algorithme glouton, ce dernier doit répéter le choix de la pièce de plus grande valeur qui ne dépasse pas la somme restante pour donner une somme avec le moins possible de pièces. Il est appelé une heuristique gloutonne dans les cas où l'algorithme ne fournit pas systématiquement la solution optimale [C. Cerrone, 2017].

5.3.4 Recuit Simulé

Le recuit simulé est un algorithme proposé par kirkpatrick en 1983 [S. Kirkpatrick, 1983]. C'est une méthode inspirée du domaine de la métallurgie, elle est caractérisée par la température T qui est une mesure importante dans le refroidissement du métal. D'après [Lebbah, 2015], l'algorithme de recuit simulé est divisé en 4 étapes suivantes (nous prenons un exemple de minimisation) :

1. Solution initiale S_0, avec une fonction d'évaluation $f(S_0)$
2. Choisir une nouvelle solution aléatoire S_1 dans le voisinage de S_0 noté par $V(S_0)$
   - Si $f(S_0) >= f(S_1)$ alors S_1 est adoptée tout en améliorant la solution courante S_0
   - Si $f(S_0) < f(S_1)$ nous choisissons un nouveau voisin de S_0 et remplacé par S_1
3. T est décrémenté lentement à chaque itération
4. Si T < € alors l'algorithme est arrêté, avec S_0 meilleure solution trouvée sinon aller à l'étape 2

L'algorithme 2 reproduit les étapes algorithmiques de la méthode de recuit simulé

Algorithm 2 Algorithme de Recuit simulé

Input S0 solution initial, K paramètre de refroidissement, € seuil fixé, T température, MaxIter nombre maximal d'itérations
Output Meilleure solution
Begin
Iter ← 0
Choisir au hasard une solution initiale S0
Repeat
  Iter ← Iter + 1
  Choisir au hasard une solution S1 dans V(S0)
  f ← f(S1) - f(S0)
  If f ≤ 0 **Then**
    S0 ← S1
  Else
    tirer au hasard P dans [0,1]
    If P ≤ e **Then**
      S0 ← S1
    End If
  End If
  T ← k * T
Until (T ≤ €) ou (Iter = MaxIter)
Return (S0)
End

5.3.5 Colonies de Fourmis

L'algorithme de colonies de fourmis fait partie des algorithmes d'intelligence en essaim, il est inspiré du comportement des fourmis lorsqu'elles cherchent de la nourriture [J. Deneubourg, 1990]. Les fourmis parviennent à trouver le chemin le plus court entre leur nid et une source de nourriture, plusieurs fourmis sortent de leur nid vers des différents points, et chaque fourmi dispose des phéromones au sol pour éclairer les autres fourmis sur le chemin qu'elle a déjà emprunté. Ainsi, lorsqu'elles reviennent au nid avec de la nourriture, les autres fourmis pourront suivre la piste de phéromones pour retrouver la source de nourriture. Les phéromones s'évaporant avec le temps et les fourmis connaîtront les chemins de la nourriture grâce à l'intensité du phéromones.

Selon le principe des fourmis, l'algorithme suggère aléatoirement plusieurs chemins, et à chaque itération il vérifie les meilleurs chemins et en propose de nouveaux. Le but de l'algorithme de colonie de fourmis est d'améliorer la solution après chaque itération jusqu'à ce que la solution optimale soit obtenue [Gardeux, 2011].

La figure (1.11) représente l'organigramme de l'algorithme colonies de fourmis.

5.3.6 Algorithme de Recherche Tabou

L'algorithme de recherche tabou a été proposé la première fois en 1986 par Glover [F. Glover, 1986]. L'idée est simple, l'algorithme ajoute chaque solution déjà visitée à une liste noire pour qu'il ne visite pas cette solution une autre fois et pour minimiser le temps de réponse. Dans les années 90, l'algorithme de recherche tabou est devenu populaire. A la manière d'une méthode par descente, le voisinage est exploré de manière complète et dès qu'une meilleure solution est trouvée, elle remplace la solution courante. Le risque dans ce type d'approche est l'apparition de cycles. C'est pour cela que la recherche Tabou possède une liste noire ou taboue afin de mémoriser les solutions déjà effectuées. Mais plus cette liste est longue, plus elle prend de place en mémoire, et de même, un temps important pour vérifier un nouveau voisin s'il existe dans la liste taboue ou non.

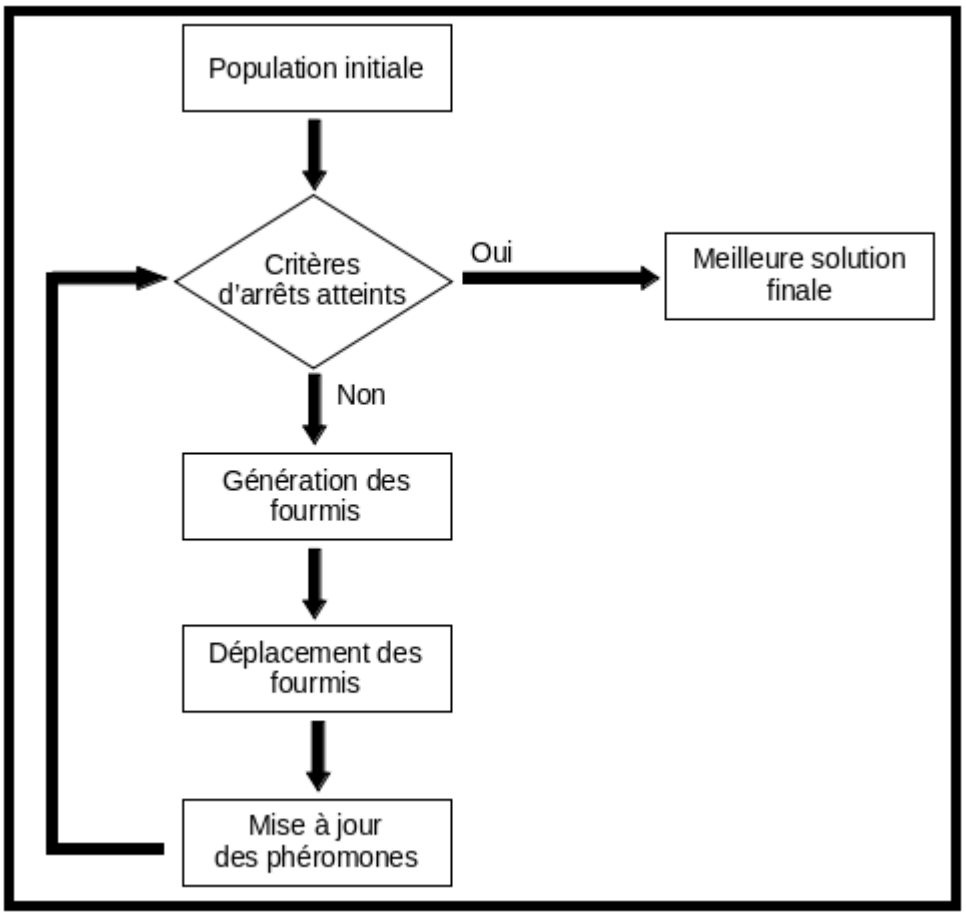


FIGURE 1.11 – Organigramme de colonies de fourmis

Pour cela, la taille de la liste taboue est devenue fixe et elle garde seulement les dernières solutions visitées. Ainsi, une solution est acceptée si le mouvement qui l'a générée n'est pas tabou [Boisson, 2008] [Lebbah, 2015].

L'algorithme 3 détaille l'algorithme classique de Recherche Tabou.

Algorithm 3 Algorithme de la Recherche Tabou

Input Solution initiale Sol
Output Meilleure solution Mei_Sol
Begin
Initialisation des mémoires à court, moyen et long terme
Repeat
  Sol ← meilleur voisin de Sol
  Mise à jour de la liste taboue
  If (critère d'intensification est vérifié) **Then**
    intensification
  End If
  If (critère de diversification est vérifié) **Then**
    diversification
  End If
Until (le critère d'arrêt soit vérifié)
Return la meilleure solution trouvée (Mei_Sol)
End

D'après [Boisson, 2008], un algorithme de recherche tabou comporte quatre types de mémoire :

a/ la mémoire à court terme : c'est une liste taboue traditionnelle pour éviter de retourner à des solutions déjà visitées. En général cette méthode tend à atteindre des solutions réalisables mais pas des solutions optimales.

b/ la mémoire à moyen terme : son but est d'atteindre des solutions optimales dans un temps acceptable. C'est la méthode la plus utilisée et la plus logique.

c/ la mémoire à long terme : son principe est de pouvoir diriger la recherche vers des solutions non encore explorées dans l'espace de recherche. Cette technique prend un temps important pour répondre et a aussi besoin d'une taille importante de mémoire. Elle est utilisée dans les problèmes compliqués et qui ne nécessitent pas une rapidité de réponse, elle a le même principe de brute force.

d/ la mémoire intelligente : Nous pouvons utilisés les trois techniques (court

terme, moyen terme, long terme) dans le même algorithme, ce dernier fonctionne par une mémoire à moyen terme, et par une fonction intelligente. Si la solution est presque optimal, l'algorithme utilise une mémoire à long terme, par contre si la solution est très faible il utilise une mémoire à court terme.

6 Conclusion

L'optimisation facilite le choix dans le cas où nous aurons une grande quantité d'alternatives possibles, elle permet de sélectionner une solution parfois optimale et parfois juste acceptable. Cependant, le plus important est qu'elle satisfait l'utilisateur dans une durée de temps minimale. L'optimisation a une relation avec la théorie des graphes puisque ceux–ci permettent de modéliser une situation ou une problématique, et la plupart des algorithmes d'optimisation peuvent être appliqués sur des graphes. Donc la relation entre la théorie des graphes et l'optimisation est une certitude.

Ce premier chapitre a effectué un état de l'art sur l'optimisation et aussi sur la théorie des graphes. Nous avons bien défini l'optimisation avec ses trois étapes (modélisation, analyse et résolution des problèmes), nous avons parlé de l'historique de l'optimisation et de la théorie des graphes où avons présenté deux types de graphes *orientés* et *non orientés*. Ce chapitre a également illustré la notion du chemin le plus court ainsi qu'un aperçu sur quelques algorithmes d'optimisation tels que les algorithmes génétiques, Dijkstra, l'algorithme A*, floyd-Warshall et les algorithmes Méta Heuristiques (gloutons, recuit simulé, colonies de fourmis, recherche tabou). Dans le prochain chapitre, nous allons présenté quelques généralités sur les systèmes de recommandations ainsi que quelques algorithmes utilisés dans la recommandation.

Chapitre 2

Systèmes de recommandation

Sommaire

1 Introduction

La croissance explosive des services sur Internet et la diversité des préférences des utilisateurs ont conduit à une difficulté à sélectionné les meilleurs services qui répondent aux besoins de l'utilisateur en termes des préférences et de contexte. De plus, les demandeurs des services sont souvent confrontés à de nombreux services concurrents qui offrent des fonctionnalités similaires.

Cependant, ils sont associés à des contraintes contextuelles différentes et ils doivent sélectionner les meilleurs dans la liste des éléments avec les fonctionnalités requises et la meilleure qualité souhaitée. Nous sommes fréquemment exposés à des situations qui nous amènent à prendre des décisions et à faire des choix, comme des films, de la musique, des articles scientifiques, des produits, des lieux de vacances, etc. La recommandation consiste à sélectionner des suggestions pertinentes en fonction des choix des utilisateurs. Les systèmes de recommandation (SR) sont développés pour anticiper les besoins des utilisateurs, et les proposer des items pertinents dans un vaste espace de ressources en fonction de leurs préférences et prédire avec précision si un utilisateur donné aimera un item [M. Mohri, 2018].

Le SR a évolué pour recommander activement les bons items aux utilisateurs en ligne, généralement sans requête de recherche explicite. Pour atteindre un tel objectif et fournir des recommandations personnalisées, un SR doit accumuler des données sur les utilisateurs et/ou les éléments disponibles, c'est-à-dire que le SR doit connaître les préférences de chaque utilisateur avant d'appliquer des techniques d'intelligence computationnelle [S. Kulkarni, 2020] (calcul statistique ou une technique bio-inspiré) pour prédire les meilleurs items. Les systèmes de recommandation personnalisés sont mis en œuvre dans de nombreuses applications et domaines commerciaux réels. Ils ont fait leurs preuves dans la musique, les réseaux sociaux, la santé, les actualités personnalisées, etc., mais semblent limités dans le tourisme contextuelle.

2 Généralités sur les systèmes de recommandation

Dans la vie quotidienne, nous sommes souvent confrontés à faire des choix, où je passe mon vacance ? Ou quelle bonne série je dois regarder, etc. Et ce n'est pas facile de recommander le meilleur choix puisque en général nous en avons plusieurs. Dans la plupart des systèmes de recommandation, le nombre des choix possibles est souvent très élevé et augmente d'une année à une autre. Par exemple, le système de recommandation de NETFLIX a enregistré 103.95 millions d'utilisateurs inscrits en juillet 2017. Cette entreprise géante aurait investi en 2017 plus de 06 Billions dollars d'après [K. Gautam, 2017] pour enrichir et améliorer son système de recommandation qui à devenue aujourd'hui un exemple pour les nouvelles entreprises du domaine. Toutes ces données montrent l'importance et l'efficacité du système de recommandation utilisé par NETFLIX et le rendement financier très important qu'il lui apporte.

2.1 Définition de la recommandation

La recommandation est une technique utilisée depuis plusieurs années, seulement, pour faciliter le choix et gagner du temps. Cette technique réduit le nombre des choix existants par la proposition d'une liste ordonnée des items pertinents à fin de satisfaire l'utilisateur [R. Burke, 2011].

Les systèmes de recommandation feront la relation entre les utilisateurs et les items. Cette relation est représentée par des mots, des commentaires, historiques de navigation, un nombre de clic ou même sur des avis positifs, neutres ou négatifs [J. Beel, 2016]. En général, il existe deux types pour étudier le principe de fonctionnement des systèmes de recommandation :

A/ Prédiction d'une note pour un utilisateur par rapport à un item qui n'a pas été déjà noté.

B/ proposition d'une liste ordonnée des items pertinents qui satisfait l'utilisateur.

2.2 Historique des systèmes de recommandation

Les systèmes de recommandation sont apparus au début des années quatre-vingt-dix pour répondre au problème de la surcharge en suggérant d'en tirer les meilleures informations d'un cumul de données. Les systèmes de recommandation sont assez proches des moteurs de recherche qui, recevant une requête de la part de l'utilisateur sous forme de texte ou d' image, répondent par une liste ordonnée qui contient soit des pages Web, images ou des vidéos, facilitant ainsi l'accès aux informations. Et ce, bien qu'ils ne reçoivent une demande de recherche, mais proposent plutôt des recommandations selon le profil, les préférences et le contexte de l'utilisateur.

En 1992 la recommandation a connu un développement dans le domaine de la recherche par la notion du filtrage d'information par la personnalisation des résultats présentés à l'utilisateur par rapport à sa requête. Elle a été ensuite utilisée dans le domaine de recherche de données et exactement dans la méthode de filtrage et l'exploitation, et cela a donné naissance aux systèmes de recommandation comme des systèmes voisins, mais indépendants des systèmes de RI. Et la différence entre les systèmes de recommandation et les systèmes de recherche d'information est présente dans le traitement de profil. Les systèmes de recommandation sont spécialisés dans le traitement des profils à long terme, alors que les systèmes RI s'intéressent aux profils à court terme [J. A. Konstan, 2012].

Tapestry est le premier système de recommandation, il a été créé par Goldberg en 1992 [D. Goldberg, 1992]. Grouplens1 et Ringo sont les deux systèmes de recommandation créés par la suite, respectivement, par [P. Resnick, 1997] et [U. Shardanand, 1995]. En 1997 Resnick et Varian ont utilisé la technique connue du filtrage collaboratif la première fois dans l'histoire et à la même année, Balavonic et Shoham ont créé le premier système de recommandation hybride associant les deux techniques les plus utilisées dans le monde, le filtrage basé sur le contenu et le filtrage collaboratif. Ils ont baptisé leur système Fab [M. Balabanovic, 1997].

D'ici, nous comprenons que le système de recommandation doit posséder des historiques sur ses utilisateurs. Plus l'historique est riche, plus le système aura

des meilleures résultats [Benouaret, 2017]. L'historique des utilisateurs peut être constitué à partir des statistiques de navigation sur internet ou les choix de leurs amis sur les réseaux sociaux voire leurs publications et commentaires.

Les systèmes de recommandation sont utilisés dans plusieurs domaines comme la culture, le tourisme, le commerce, et même la médecine où le système recommande des conseils de bien-être. La figure suivante représente la page d'accueil du site d'Amazon qui contient un système de recommandation très efficace et qui attire des clients de chaque bout du monde pour consulter, proposer et acheter des millions des produits et services.

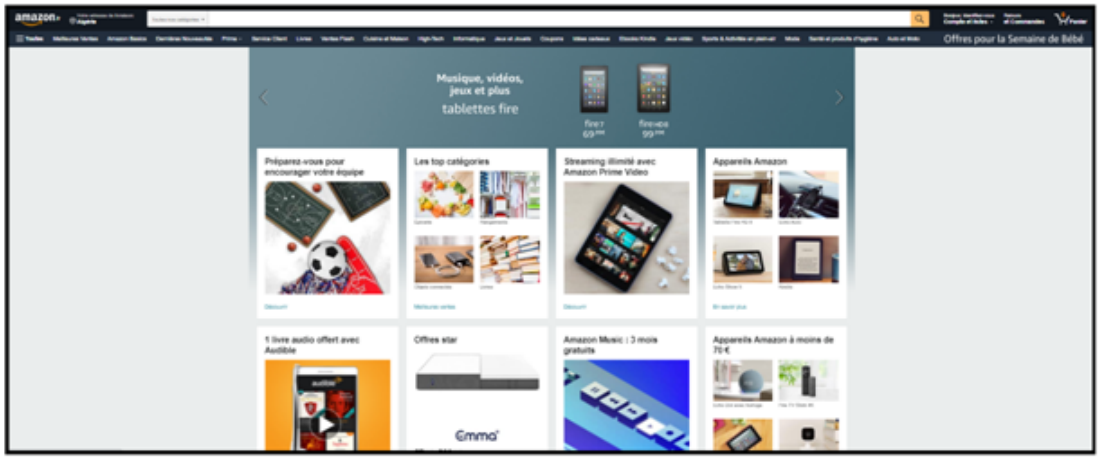


FIGURE 2.1 – Page d'accueil du site Amazon

3 Approches basées sur le contenu

Les méthodes de recommandation basées sur le contenu utilisent l'analyse des contenus et les informations existants sur les items. C'est presque comme la recherche d'information où nous cherchons des documents pertinents pour une requête donnée. La recommandation basée sur le contenu ne nécessite pas cependant une requête mais se base sur la similarité du profil de l'utilisateur avec d'autres profils ainsi que sur la liste des préférences de l'utilisateur. Aussi, nous notons que les nouveaux items peuvent être recommandés dans les systèmes de recommandation basés sur le contenu, par contre c'est rare de recommander des

nouveaux items dans un système de recommandation basés sur une approche de filtrage collaboratif.

3.1 Définition du filtrage basé sur le contenu

Le filtrage basé contenu exploite seulement les représentations des informations des items qui existent, et il se base sur le degré de la similarité entre des items dans une matrice termes-items, pour avoir un degré de pertinence pour un item donné, c'est à dire si un utilisateur exprime un intérêt pour un item, alors tous les items similaires seront jugés potentiellement pertinents aussi mais avec un degré de pertinence [Lang, 1995] [Villalobos, 2014]. Les SR basées contenu infèrent plutôt les préférences de l'utilisateur et lui recommandent les items similaires aux items qu'il a aimés auparavant. Ainsi, quand de nouveaux items sont introduits dans le système, il peuvent être recommandés directement, sans que cela ne nécessite un temps d'intégration comme c'est le cas pour les SR basés sur le filtrage collaboratif [M. Pazzani, 1997] [Benouaret, 2017].

3.2 Différentes approches basées sur le contenu

Il existe plusieurs approches basées sur le contenu et la différence entre ces celles – ci est représentée par le type ou la méthode de la similarité.

3.2.1 Recommandation basée sur la connaissance

Cette méthode consiste à collecter une grande quantité d'informations sur l'utilisateur pour en avoir un profil assez précis et, ainsi, la recommandation ne peut être que très proche et exacte de ses besoins et / ou préférences. Un système basé sur le contenu (knowledge-based) n'est pas confronté au problème de démarrage à froid. Autrement dit, par exemple, si on veut recommander un appartement pour l'achat à un utilisateur, il est nécessaire d'avoir quelques informations précises de l'utilisateur comme le nombre des pièces de l'appartement qu'il souhaiterait acquérir ou le montant maximal qu'il est prêt à dépenser pour cet éventuel achat. Avec ces quelques informations, on évite ce qu'on appelle le

problème du démarrage à froid [H. Bouhissia, 2021].

3.2.2 Recommandation basée sur l'utilité

Un système de recommandation basé sur l'utilité (utility-based) fait des recommandation par le calcul de l'existence de chaque objet pour un utilisateur donné, par la création d'une fonction d'utilité. La plus simple façon d'avoir des informations de cet utilisateur est de lui demander ensuite de remplir un formulaire d'inscription. A partir de ces informations, nous pouvons lui recommander, à titre d'exemple, l'utilisation d'un transport en se référant, bien entendu, à son adresse. Ainsi, s'il habite près d'une gare, l'utile est de lui proposer un abonnement d'une année avec un prix attractif [Z. Fayyaz, 2020].

3.2.3 Recommandation basée sur les vecteurs de mots-clés

La recommandation basée sur les vecteurs de mots-clés utilise le modèle VSM (Vector Space Model). Le système présente vectoriellement des informations sur l'utilisateur principal pour réaliser une similarité avec les autres utilisateurs qui existent déjà dans la base de données afférente. La similarité est généralement calculée par la formule connue du TF-IDF (Terme Frequency-Inverse Document Frequency), ou par d'autres formules comme distance de Jaccard, Cosinus ou Distance de Manhattan [Zhenyu, 2012].

3.3 Avantages du filtrage basé sur le contenu

Le filtrage basé sur le contenu a plusieurs avantages, il peut répondre aux intérêts à long terme des utilisateurs, en employant des techniques efficaces dans le domaine de l'intelligence artificielle pour la mise à jour des profils et le recoupement entre profils et items. En outre, l'utilisateur dans un tel système ne dépend absolument pas des autres. Ainsi, il recevra des recommandations du système même s'il est le seul inscrit, pour peu qu'il ait décrit son profil en donnant un ensemble de thèmes qui l'intéressent.

3.4 Inconvénients du filtrage basé sur le contenu

La technique de filtrage basé sur le contenu est soumise à l'effet « entonnoir », car le profil évolue naturellement par restriction progressive sur les thèmes recherchés. Ainsi, l'utilisateur ne reçoit que les recommandations relatives aux thèmes présentés dans son profil, une fois devenu stable. Par conséquent, il ne peut pas découvrir de nouveaux domaines potentiellement intéressants pour lui. Le filtrage basé sur le contenu est également victime d'un effet de masse, car ne bénéficiant pas des jugements de qualité que d'autres utilisateurs ont pu faire sur les items qu'il reçoit, c'est l'utilisateur lui-même qui devra procéder à l'écrémage des items reçus, écrémage qui fait intervenir d'autres critères que celui de la thématique. Le filtrage basé sur le contenu doit également faire face au problème du démarrage à froid. Par exemple, un nouvel utilisateur rencontre plus ou moins de difficultés à définir les thèmes qu'il préfère, afin que le système instance son profil, malgré certaines techniques d'apprentissage permettant d'en inférer à partir des textes « exemples » fournis par l'utilisateur.

4 Approches basées sur le filtrage collaboratif

Un système de recommandation basé sur le filtrage collaboratif est un système qui repose sur le partage de l'opinion entre les différents utilisateurs.

4.1 Définition du filtrage collaboratif

L'approche basée sur le filtrage collaboratif a été définie et utilisée dans plusieurs travaux comme ceux de [A. Kulkarni, 2019] et [A. Umanetsa, 2014]. Plusieurs définitions sont d'ailleurs données. Mais, pour résumer, le principe de cette approche est le suivant : le système choisit des utilisateurs similaires (qui ont des goûts ou des préférences similaires) à l'utilisateur principal pour recommander une liste des items jugée positive par les utilisateurs similaires mais qui n'a pas été votée ou notée par l'utilisateur principal (de nouveaux items). Si on prend par exemple un utilisateur principal qui a une liste d'amis (amis, voisin ou collègue, etc.), dans cette liste, il existe une autre liste des amis similaires (qui ont

les mêmes goûts ou préférences) et qui aiment regarder pratiquement les mêmes films regardés par l'utilisateur principal (dans ce cas le système de recommandation doit recommander cette liste des films). Partant de cette information, si un nouveau film est parfait pour les utilisateurs similaires alors automatiquement sinon probablement l'utilisateur principal va également l'adorer et le système doit le recommander. Dans le cas où le film déplaît aux utilisateurs similaires, le système choisit automatiquement de ne pas le recommander [G. Adomavicius, 2005] [Maatallah, 2016].

4.2 Différentes approches basées sur le filtrage collaboratif

Il existe plusieurs approches, basées sur le filtrage collaboratif, qui partagent le même concept avec quelques points de différence, nous présentons ici les méthodes suivantes :

4.2.1 Filtrage collaboratif basé sur les voisins

K plus proche voisin KNN (k-Nearest Neighbours) est l'algorithme le plus utilisé dans les approches de filtrage collaboratif basées sur les voisins. Le système essaie d'identifier les meilleurs voisins pour un utilisateur donné, en traitant la matrice des évaluations dont chaque ligne correspond à l'historique des évaluations d'un utilisateur. Plus précisément, étant donné l'utilisateur U, l'approche des voisins les plus proches se réalise en général en deux étapes :

E1. Mesurer la (dis)similarité entre l'utilisateur principal U et les autres utilisateurs.

E2. Sélectionner les meilleurs voisins en fonction de la (dis)similarité entre l'utilisateur U et les autres calculée dans l'étape précédente.

Enfin, afin de sélectionner les meilleurs voisins pour l'utilisateur U dans l'étape E2, il existe deux stratégies possibles. D'abord, nous pouvons utiliser un seuil pour la proximité entre utilisateurs. Cette méthode permet de contrôler la qualité des items proposés, mais le nombre des items risque d'être très faible pour calculer la prédiction, si le seuil de proximité est trop fort. Une autre alternative est

d'utiliser un seuil K pour fixer la taille maximale de l'ensemble des voisins (K voisins les plus proches).

En résumé, l'approche des voisins les plus proches exploite des relations explicitement disponibles dans les profils des utilisateurs. Elle est simple à implémenter, et efficace dans la plupart des cas. Mais, cette approche souffre du coût de calcul, car le système doit traiter à chaque fois la matrice entière des évaluations afin de former des communautés [BenTicha, 2015].

4.2.2 Les modèles probabilistes

Les méthodes probabilistes utilisent des algorithmes de probabilités et statistiques. Ces derniers ont toujours une relation avec l'aléatoire, c'est-à-dire ce type d'algorithme utilise quelques fonctions d'aléatoire dans quelques étapes de déroulement. Pour cela, nous pouvons avoir deux résultats différents si nous appliquons le même algorithme probabiliste deux fois sur le même jeu de données, car il utilise des valeurs aléatoires qui changent d'un résultat à un autre. Le but des modèles probabilistes est de pouvoir prédire les comportements d'un utilisateur au future, les algorithmes calculent la probabilité de si un utilisateur va aimer un item ou non. Sachant que l'item n'est pas visité et noté par l'utilisateur. Les bons systèmes de recommandation probabilistes sont les systèmes qui retournent les probabilités de la réalité [Caussinus, 1993].

4.3 Avantages du filtrage collaboratif

Les systèmes de recommandation basés sur le filtrage collaboratif comme tous les systèmes ont des avantages et des inconvénients, le système basé sur le filtrage collaboratif peut utiliser les scores d'autres utilisateurs pour évaluer l'utilité des éléments, il à aussi comme avantage la possibilité de trouver des utilisateurs ou groupes d'utilisateurs dont les intérêts correspondent à l'utilisateur courant, et plus qu'il existe des utilisateurs plus qu'il existe du scores : meilleurs sont alors les résultats.

4.4 Inconvénients du filtrage collaboratif

Les systèmes de recommandation basés sur le filtrage collaboratif ont une difficulté pour trouver des utilisateurs ou groupes d'utilisateurs similaires. Aussi le système de recommandation se heurte à la faible densité des informations des utilisateurs, de plus, il existe aussi le problème du démarrage à froid (cold-start problem), lorsqu'un nouvel utilisateur utilise le système, ses préférences ne sont pas connues et lorsqu'un nouvel élément est ajouté au catalogue, personne ne lui a attribué de score, dans les systèmes avec un grand nombre d'éléments et d'utilisateurs, le calcul croît linéairement, des algorithmes appropriés sont donc nécessaires.

5 Approches hybrides dans les systèmes de recommandation

Les approches hybrides combinent deux ou plusieurs approches de recommandation. La combinaison entre les différentes approches permet de bénéficier des avantages des approches utilisées et d'éviter les inconvénients. Le but étant d'améliorer le degré de pertinence des systèmes. Par exemple, le démarrage à froid au niveau des items des approches basées sur le filtrage collaboratif peut bénéficier des avantages des approches basées sur le contenu. En effet, dans une approche basée sur le contenu, seule la description de l'item est utilisée dans la recommandation, il n'y a donc pas besoin que l'item soit noté par un certain nombre d'utilisateurs avant de pouvoir être recommandé, comme c'est le cas dans les approches basées sur le filtrage collaboratif.

L'hybridation a donné dernièrement des bons résultats en matière de recherches, et plusieurs chercheurs se sont d'ailleurs focalisés sur cette technique pour améliorer les différents systèmes de recommandation ainsi que la recherche d'information et les moteurs de recherches et d'autres systèmes. Il existe plusieurs types d'hybridations dans les systèmes de recommandation, nous citons :

5.1 Hybridation pondérée

D'après [B. Mobasher, 2004], L'hybridation pondérée est la plus simple, puisque elle consiste à calculer le score de chaque item à recommander par l'utilisation d'une fonction qui combine entre deux ou plusieurs systèmes de recommandation. La combinaison du score peut être calculée par n'importe quelle formule ou par des pourcentages associés à chaque système. Le nouveau score obtenu présente le résultat du nouveau système de recommandation hybride. [Lemdani, 2016] a présenté un schéma explicatif de l'hybridation pondérée dans la figure suivante :

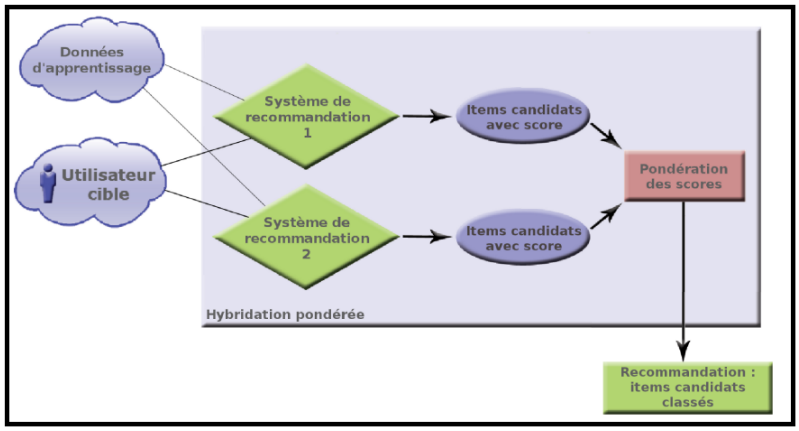


FIGURE 2.2 – Schéma explicatif de l'hybridation pondérée

5.2 Hybridation à bascule

Le système dans l'hybridation à bascule sélectionne une approche de recommandation plutôt qu'une autre, selon les résultats obtenus. C'est-à-dire si les résultats obtenus par le premier système sont insuffisants et faibles, le système utilise alors automatiquement une autre approche. Cette hybridation doit être testée sur un échantillon avant d'être exécutée sur tout l'ensemble. Donc le système va pouvoir alterner entre deux systèmes de recommandation différents. Cette hybridation est presque similaire à celle de la pondération, à la différence que dans l'hybridation pondérée, chaque système renvoie une liste des items à recommander avec un degré (l1 et l2) puis le système fait une combinaison entre les deux listes. En revanche dans la technique de l'hybridation à bascule, une

procédure de sélection des produits selon certains critères sera effectuée, et le résultat sera soit I1, soit I2 et non pas une combinaison entre les deux listes comme le cas de l'hybridation pondéré [A. Kenaan, 2020].

5.3 Hybridation par l'ajout de caractéristiques

L'hybridation par l'ajout de caractéristiques permet de combiner deux ou plusieurs systèmes de recommandation, le premier calcule un attribut ou un ensemble d'attributs, qui seront utilisés par le second et ainsi de suite s'il existe d'autres systèmes dans l'hybridation. L'hybridation par l'ajout de caractéristiques est une technique sensible à l'ordre car le deuxième modèle utilise uniquement la sortie du premier système. et il est impossible de changer l'ordre des systèmes. La caractéristique ajoutée par un système permet d'utiliser des astuces que l'autre système n'aurait pas pu générer. Cette hybridation fait la relation entre les systèmes utilisés par l'utilisation du résultat des approches comme des paramètres dans les approches suivantes. Par exemple, une première approche de recommandation est sensible aux profils des utilisateurs afin de générer un ensemble d'items. Et la deuxième approche de recommandation est sensible aux préférences des utilisateurs. La deuxième approche doit prendre les résultats de la première approche comme paramètres pour enrichir la qualité de la recommandation [P. Melville, 2002]. La figure (2.3) schématise le fonctionnement de cette hybridation.

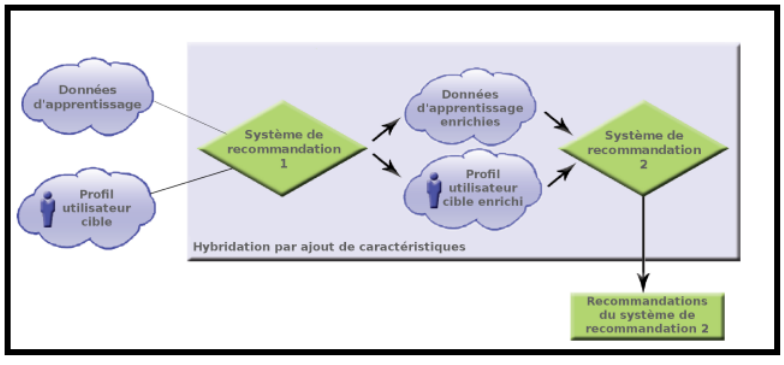


FIGURE 2.3 – Schéma explicatif de l'hybridation par l'ajout de caractéristiques

Il existe d'autres types d'hybridation dans le domaine des systèmes de recom-

mandation comme l'hybridation mixée, l'hybridation par combinaison de caractéristiques, l'hybridation en cascade et l'hybridation méta-niveau.

6 Limites constatées des systèmes de recommandation

Le problème de démarrage à froid est l'un des problèmes connu dans la recommandation. Il empêche le système de faire des recommandations pour les nouveaux utilisateurs car il est impossible de proposer quelque chose pour une personne dont on ignore l'identité et sur laquelle nous n'avons aucune information. Dans ce cas, le système ne dispose pas d'assez d'informations à son sujet voire parfois aucune donnée. Nous pouvons aussi avoir un démarrage à froid sur les items. Dans le cas où le système reçoit un nouvel item mais avec peu d'informations et de données, le système ne va pas le bien recommander. Pour régler ce problème, il faudrait utiliser des algorithmes de pré-traitement des informations de la base de données du système ou même d'ignorer totalement les items qui manquent d'informations voire de calculer les mesures de similarité entre les utilisateurs pour construire un profil d'un utilisateur ou pour enrichir les informations d'un item. Par exemple, si l'utilisateur est un enfant, nous supposons, en général, qu'il a les mêmes caractéristiques que celles des autres enfants se trouvant dans la base de données, ce qui nous permettra d'enrichir le profil de l'enfant par d'autres informations et d'autres préférences similaires à celles des autres enfants [Gras, 2018].

Il existe d'autres limites pour les systèmes de recommandation comme le changement de comportement des utilisateurs car plusieurs utilisateurs peuvent avoir plusieurs goûts d'après le temps ou d'après leurs places, par exemple dans le mois sacré de Ramadan, la majorité des musulmans n'écoutent pas la musique, par contre un système de recommandation ne prend pas ce comportement en considération.

La Sur-spécialisation est un problème qui concerne les systèmes qui recommandent à un utilisateur que des items qui sont en relation avec son profil, dans

ce cas l'utilisateur est limité aux items similaires aux items qui a déjà aimées. En-revanche, la diversité des recommandations est souvent une caractéristique souhaitable pour les systèmes de recommandation.

La parcimonie est aussi un problème connue dans la recommandation, par exemple si un item qui a reçu peu d'avis a moins de chances d'être recommandé par rapport aux autres items, c'est a dire si le nombre de notes déjà obtenues est très faible par rapport au nombre de notes qui doivent être prédites. Et la même chose pour un utilisateur qui a noté des items qui n'ont pas reçu beaucoup d'avis, il est difficile de lui trouver des utilisateurs similaires, du coup il sera difficile de faire des recommandations pertinentes pour cet utilisateur, ce problème concerne seulement les SR basés sur le filtrage collaboratif [Benouaret, 2017].

7 Quelques algorithmes utilisés dans la recommandation

En recommandation, il existe plusieurs et divers algorithmes, chacun a ses caractéristiques, nous présentons ici quelques algorithmes connus en recommandation :

7.1 Classificateur naïve bayésienne

Le classificateur naïve bayésienne est un classificateur probabiliste simple basé sur le théorème de Bayes, ayant une précision de classification élevée lorsqu'il traite une grande quantité de données. Il utilise les informations de l'utilisateur principal pour proposer ou attribuer une classe à partir d'une liste de classes déjà indiquées. Il existe deux types de fonctionnement de l'algorithme naïve de Bayes, le filtrage collaboratif et le filtrage basé sur le contenu. Le filtrage collaboratif consiste à faire une similitude entre les utilisateurs pour recommander les informations les plus similaires, mais pour le filtrage basé sur le contenu, au contraire, on calcule la similitude entre les items (article, film, lieu touristique, etc.) et nous recommandons les meilleurs items à l'utilisateur [L. Shuxian, 2019].

Nous présentons le classificateur naïve bayésienne dans l'algorithme 4.

Algorithm 4 Classificateur Naïve Bayésienne

Input
Ensemble de données d'entraînement
Informations du nouveau utilisateur
N : Nombre de classes
M : Nombre de colonnes
Output
Classe du nouveau utilisateur
Begin
For i in range (0,N)
For j in range (0,M)
calculer l'espérance de la colonne j de la classe i
calculer la variance de la colonne j de la classe i
End For
End For
évidence $\leftarrow \sum_1^N (P(classe) \sum_1^M P(colonne/classe))$
For i in range (0,N)
P(classe) $\leftarrow$ (P(classe) $\sum_1^M P(colonne/classe))/évidence$
End For
Return (Classe qui a la meilleure probabilité)
End

7.2 Forêt aléatoire

La forêt aléatoire est un algorithme essentiel qui fait partie des algorithmes d'apprentissage automatique. Il s'agit d'un algorithme basé sur l'assemblage d'arbres de décision. Il est assez intuitif à comprendre, rapide à entraîner et il produit des résultats généralisables. Il est basé sur l'aléatoire mais aussi sur des arbres de décision, c'est un algorithme itératif de la famille des Set Techniques algorithmes d'analyse prédicative. On note que l'algorithme de bagging ainsi que l'algorithme de boosting sont de cette famille citée plus haut et la différence est qu'en forêt aléatoire, les arbres sont individuellement efficaces et de grande profondeur [B. S. Abdualgalil, 2020]. Le code suivant représente l'algorithme de forêt aléatoire.

Sachant que :

Algorithm 5 Forêt aléatoire

Input Données de modèle (Td)
Output Classe
Begin
For i in range (1,t)
  Sélectionné Tdi depuis les données de modèle Td au hasard
  Généré un nœud racine appelé Rni par l'utilisation de Tdi
  Ni[i] = GenerateTree(Rni)
End for
If Ni consiste en une seule instance de classe **Then**
  Return (Classe)
Else
  Fractionner des caractéristiques dans Ni de manière aléatoire
  For j in range (1,Nb-classes)
    Calculer le gain de la classe j
  End for
  Return (Meilleure Classe d'après le gain)
End If
End

Td : Training Data

t : nombre des arbres a généré

Rni : Root Node

N : node

7.3 Réseau neuronal

Réseau neuronal est l'un des algorithmes les plus connus dans le domaine de l'informatique, très utilisé dans plusieurs domaines tels que le traitement du langage, l'optimisation, la classification et également dans les systèmes de refonte. Il donne de bons résultats dans le système de remplissage et surtout utilisé par plusieurs systèmes connus dans le monde.

Un réseau de neurones artificiels s'inspire du fonctionnement des neurones biologiques, et qui s'est par la suite rapproché des méthodes statistiques. Les réseaux de neurones sont généralement optimisés par des méthodes d'apprentissage de type probabiliste et d'autre part dans la famille des méthodes d'intelligence

artificielle [A. Kulkarni, 2019]. Dans la modélisation des circuits biologiques, ils permettent de tester certaines hypothèses fonctionnelles issues de la neurophysiologie ou les conséquences de ces hypothèses, afin de les comparer à la réalité [k. Domokos, 2019].

7.4 K-plus proches voisins

La méthode K Nearest Neighbor (KNN) a été largement utilisée dans les applications d'exploration de données et d'apprentissage automatique en raison de sa mise en œuvre simple et de ses performances remarquables. Cependant, il a été prouvé que la définition de toutes les données de test avec la même valeur K dans les méthodes KNN précédentes rend ces méthodes peu pratiques dans les applications réelles. KNN est un classificateur entre deux ou plusieurs classes, très utilisé dans la recommandation, KNN peut utiliser plusieurs formules de distance comme la distance euclidienne, Manhattan, Minkowski ou Tchebychev. Les méthodes KNN sont également utilisées pour la régression et l'imputation des données manquantes, d'exploration de données, et la classification. Le classificateur KNN a montré des performances remarquables sur des données de grande taille. Cependant, les performances de la classification KNN peuvent être affectées par certains problèmes, tels que la sélection de la valeur K, la sélection des mesures de distance, etc. Il existe de nombreuses techniques récentes qui font partie de l'algorithme KNN avec quelques améliorations comme KNN-CF (facteur de certitude), HKNN K-local Hyperplane Distance Nearest Neighbor et K trajectoires voisines les plus proches (K-NNT) [Y. Cai, 2010] [S. Zhang, 2017].

7.5 Régression logistique

La régression logistique est une modélisation d'une variable dépendante (que l'on veut prédire ou expliquer) qualitative dichotomique : sains/malades, exposés/non exposés. Si les variables sont dépendantes à plusieurs modalités, nous dirons que nous sommes dans le cas de la régression polytomique. La technique de la régression logistique permet d'étudier la relation entre une variable binaire et des variables qualitatives et quantitatives. Elle permet aussi de traiter les cas où

la variable à expliquer possède plus de deux classes si celles-ci ne peuvent pas être ordonnées, puisque l'ordre dans la recommandation a une importance [Bouyer, 2020] [A. Gillet, 2011].

7.6 Classificateur d'amplification de gradient

L'algorithme de gradient consiste à construire un modèle pour prédire des classes. Il est connu par le nom amplification de gradient (en Anglais Gradient Amplification) et utilisé dans le domaine de Deep Learning. Il représente les items comme des vecteurs, chaque item est représenté par plusieurs points. Le gradient contient une fonction de voisinage qui détermine les classes proches. Cette fonction de voisinage est différentielle. Le gradient est la généralisation à plusieurs variables de la dérivée d'une fonction d'une seule variable. Le gradient a une importance capitale en recommandation, analyse vectorielle et en physique, où il fut d'abord employé. Utilisé en théorie des variations, il est aussi fondamental dans le domaine de l'optimisation ou de la résolution d'équations aux dérivées partielles [S. Basodi, 2020].

C'est l'un des algorithmes qui gagne en popularité en raison de sa vitesse et de sa précision de prédiction élevées, principalement lorsqu'il s'agit d'ensembles de données volumineux et compliqués. Le gradient est l'un des algorithmes de boosting, utilisé pour minimiser l'erreur de biais du modèle, sachant que les erreurs sont généralement classées en deux catégories, à savoir l'erreur de biais et l'erreur de variance [BenFadhel, 2019].

8 Quelques métriques d'évaluation utilisées

Les mesures ou les métriques d'évaluation mesurent la capacité de l'algorithme de recommandation à prédire correctement les évaluations connues des utilisateurs. Pour évaluer un système, nous devrons avoir une partie test qui contient deux colonnes :

• La vraie classe : cette colonne contient la classe associée à une ligne donnée et se trouve dans la base de données par défaut.

• La classe prédite qui représente le résultat obtenu par le système.

Les métriques comparent les vrais résultats avec les résultats récupérés par le système qui utilise la partie modèle pour prédire les classes de la partie test. Si le système a prédit correctement un nombre important des classes, il est parfait et, dans le cas contraire, il serait faible. La plupart de ces métriques proviennent du domaine de la recherche d'information, nous présentons dans les prochaines sous sections les meilleures mesures de performance.

8.1 Précision

Elle représente la relation entre le nombre d'items pertinents sélectionnés et l'ensemble des items sélectionnés. Nous donnons ici deux exemples pour mieux comprendre l'importance de la précision. Pour chacun des deux exemples, la partie test contient 5 items pertinents.

A/ Exemple où la précision est faible.

Si le système récupère les 5 items pertinents et 15 autres items non pertinents, dans ce cas la précision est de l'ordre de 5 sur 20 c'est-à-dire 0.25, la précision est donc faible.

B/ Exemple où la précision est parfaite.

Pour le second exemple, le système a récupéré uniquement 3 items, mais tous sont pertinents. Dans ce cas, la précision est de 100% et, donc parfaite.

La précision représente la possibilité qu'un item sélectionné soit pertinent. Voir la formule (2.1) :

$$Précision = \frac{NIPS}{NTIS} \tag{2.1}$$

Sachant que NIPS représente le Nombre des Items Pertinents Sélectionnés par le système, et NTIS représente le Nombre Total des Items Sélectionnés [Haydar, 2014].

8.2 Rappel

C'est une mesure très utile dans l'évaluation des systèmes de recommandation, elle représente le rapport entre le nombre d'items pertinents choisis par le système et le nombre total des items pertinents disponibles dans la partie test de la base de données. La mesure de rappel représente la probabilité qu'un item pertinent soit sélectionné. La formule suivante représente la formule utilisée dans notre projet pour calculer le rappel :

$$Rappel = \frac{NIPS}{NIPD} \tag{2.2}$$

Sachant que NIPS représente le Nombre des Items Pertinents Sélectionnés par le système et NIPD représente le Nombre Total des Items Pertinent Disponible [D. Werner, 2014].

8.3 F-mesure

La métrique F-Mesure est une combinaison entre le rappel et la précision [Tadlaoui, 2018], elle est définie par la formule suivante :

$$F - mesure = \frac{2 * Rappel * Précision}{Rappel + Précision} \tag{2.3}$$

8.4 Accuracy

C'est aussi une mesure importante qui aide à mesurer la performance d'un système. Elle représente le pourcentage des items pertinents qui existe, c'est la relation du nombre total des items pertinents dans la partie test par rapport au nombre total des items de la partie test. Elle est définie par la formule suivante :

$$Accuracy = \frac{NTIP}{NTI} \tag{2.4}$$

Sachant que NTIP représente le Nombre Total des Items Pertinents, et NTI représente le Nombre Total des Items [Michailidis, 2017].

8.5 Erreur Absolue Moyenne

MAE Erreur Absolue Moyenne (Mean Absolute Error) est une méthode de mesure statistique utilisée pour mesurer la qualité de la recommandation. MAE calcule l'écart entre les notes prédites et les notes réelles sur les utilisateurs de test, indiqué dans l'équation suivante :

$$MAE = \frac{\sum_1^N |Pij - Aij|}{N} \tag{2.5}$$

Ou,

N - Le nombre total d'éléments prédits

Pij- La valeur de notation prédite pour l'utilisateur i sur l'élément j

Aij - La notation réelle de l'utilisateur i sur l'élément j

Si la valeur MAE est faible, le système prédit des évaluations précises des utilisateurs. Cela fournira une bonne qualité de recommandation [T. Chai, 2014].

8.6 Racine de l'Erreur Carré Moyenne

RMSE (Root Mean Square Error) et en francais (Racine de l'Erreur Carré Moyenne) est une règle qui mesure l'amplitude moyenne de l'erreur, de sorte que la différence entre les prédictions et les valeurs observées correspondantes sont mise au carré [A. Odic, 2013] [T. Chai, 2014], la formule suivante représente la formule de RMSE :

$$RMSE = \sqrt{\frac{\sum_1^N (pred(u,i) - V(u,i))^2}{|N|}} \tag{2.6}$$

Sachant que,

- pred(u,i) représente la note prédite par le système pour l'utilisateur U et l'item I.

- V(u,i) représente la note réelle qui a donné l'utilisateur U à l'item I.

- N représente le nombre total des notes.

Un système de recommandation idéal est un système qui sélectionne tous les items pertinents (rappelle = 1), et tous les items qu'il retrouve sont pertinents (précision=1). En pratique, ce seuil idéal n'est jamais atteint puisque ces deux mesure évoluent en sens inverse. Intuitivement, si nous augmentons le rappel en retrouvent plus d'items pertinents, et si nous diminuons la précision en retrouvant aussi plus d'items non pertinents, la mesure de f-mesure a une importance de combiner entre le rappel et la précision et un bon système c'est celui ayant une mesure de f-mesure très élevée.

La MAE et la RMSE ne peuvent pas être calculées que pour les algorithmes de recommandation qui prédisent la valeur du vote d'un item, et plus la MAE ou la RMSE est faible, plus le système est efficace.

9 Profils, préférences et contextes

9.1 Notion du profil d'un utilisateur

Un profil est créé lorsque les utilisateurs souhaitent personnaliser le comportement d'un service selon leurs besoins, à l'exemple de proposé un parc d'attraction si l'utilisateur est un enfant, par contre on propose une salle de cinéma c'est l'utilisateur est un adulte. La majorité des systèmes de recommandation sont sensible aux profils des utilisateurs, c'est-à-dire ils fonctionnent d'après le profil de l'utilisateur. Par exemple, si nous prenons deux personnes qui ont les mêmes informations (âge, sexe, nationalité, langue, etc.) mais qui ne partagent pas le même domaine du travail, dans ce cas, la recommandation va être différente, car ils n'ont pas le même profil.

Dans [Barbosa, 2014], l'auteur à transféré les profils des utilisateurs sous une dimension plus petite à l'aide de SVD, et il a clusterné les profils à l'aide d'une version modifiée de l'algorithme k-means. Pour chaque cluster, un profil utilisateur est publié, où chaque poids dans le profil utilisateur est représenté par la valeur moyenne donnée par les membres du cluster. Cette comparaison des profils aide à les enrichir.

Le profil aide l'utilisateur à personnaliser son interface et aide le système à lui offrir le meilleur service qui le satisfait.

9.2 Notion des préférences d'un utilisateur

Les préférences ont un rôle très important dans les systèmes de recommandation car chacun de nous à ses propres goûts et ses centres d'intérêt. Le système doit donc être attentif aux préférences des utilisateurs pour les satisfaire par des items qui les intéressent. Les préférences changent d'un système à un autre selon le domaine de la recommandation, par exemple, dans un système de recommandation des films, nous trouvons plusieurs catégories de films (comédie, action, film de guerre, etc.) mais aussi la préférence à considérer, tout comme les autres points, à l'exemple de l'année de la sortie du film. Parmi les utilisateurs, il y a ceux qui préfèrent les anciens films et d'autres les tous nouveaux.

Dans [K. Lakiotaki, 2010], les auteurs ont travaillé beaucoup plus sur les préférences des utilisateurs. Aussi, leur système recommande des items selon les préférences, mais il calcule aussi la similarité entre les utilisateurs sur la base des préférences, d'où l'importance de celles–ci dans les systèmes de recommandation.

Les préférences est la liste des services ordonnés par rapport à l'importance d'un service à un autre pour l'utilisateur. Autrement dit, ce dernier préfère un tel service à un autre.

9.3 Notion du contexte d'un utilisateur

Nous avons parlé dans les deux paragraphes précédents de l'importance du profil et des préférences des utilisateurs dans les systèmes de recommandation. Cependant la sensibilité au contexte n'est pas de moindre importance lorsqu'il s'agit de satisfaire l'utilisateur.

La sensibilité au contexte représente la capacité d'un système à découvrir et à réagir à l'état de l'environnement mais aussi l'adaptation au changement de ce contexte, qui a une duré de vie et change certes avec le temps. Tous simplement la sensibilité du contexte représente la prise en compte du contexte d'utilisation. En

effet, un système sensible au contexte doit percevoir la situation de l'utilisateur, dans son environnement et adapter son comportement à la situation, pour que ce dernier soit satisfait [M. F. Khalfi, 2014].

9.3.1 Définition de la notion de contexte

Il existe plusieurs définitions sur la notion de contexte, il y'a tout d'abord celle de Manuelle Kirsch qui définit le contexte par "Un système sensible au contexte peut être considéré comme un système qui est capable de détecter, à un instant, un ou plusieurs éléments du contexte dans lequel ou lesquels se trouve l'utilisateur et de fournir, en retour, soit des informations, soit des services adaptés à ce contexte et aux changements d'un ou de plusieurs éléments afférents. L'évolution de ces systèmes dépend donc de la compréhension et de la maîtrise de la notion de contexte. Définir ce qu'on entend par contexte est une question complexe" [M. Kirsch, 2007].

Salber et Dey abowd ont aussi définit le contexte par "les informations environnementales ou le contexte couvrent les informations qui font partie de l'environnement d'exploitation d'une application et qui peuvent être détectées par l'emplacement, l'identité, l'activité et l'état des personnes, des groupes et des objets." [D. Salber, 1999].

9.3.2 Catégories du contexte

Il existe plusieurs contextes dans la recommandation qui différent d'un domaine à un autre. Par exemple dans le domaine du tourisme, le contexte représente la température, l'état de la météo, la distance, la localisation, etc. tandis que pour le domaine de la recommandation des films, le contexte représente par exemple le pays de l'utilisateur. Ainsi, s'il est connecté depuis l'Italie, le système lui recommandera des films sous titrage italien et s'il est dans un pays arabe, le système va lui recommander beaucoup plus des films traduits en arabe.

Benouaret [Benouaret, 2015] a créé un système de recommandation des musées qui tient compte de cinq éléments :

Individualité : Elle repose sur les informations des utilisateurs (âge, niveau

d'expertise, etc.). Deux utilisateurs sont similaires d'après leurs informations, ils auront tendance à avoir les mêmes préférences.

Localisation : La constitution du parcours dans le musée est réalisée d'après la localisation de l'utilisateur ainsi que l'emplacement des œuvres d'art. Le parcours est constitué pour satisfaire les préférences de l'utilisateur et en même temps pour réduire le temps de visite.

Temporalité : L'utilisateur choisit le temps qu'il veut passer au musée et devant chaque œuvre. Grâce aux informations obtenues sur ces deux contextes, le système peut calculer le nombre des œuvres à visiter pendant ce parcours.

Activité : Chaque utilisateur a une liste des oeuvres qu'il a déjà visitées et notées. Le système lui propose des nouvelles œuvres qui ont les mêmes caractéristiques que celles qu'il aurait appréciées.

Relations : Le système recommande aussi des œuvres qui sont visitées et bien notées par les utilisateurs similaires de l'utilisateur principal.

Le contexte peut être divisé en deux classes :

- Contexte statique :

Contient des informations qui ne changent pas avec le temps comme, le genre, date de naissance, et la langue.

- Contexte dynamique :

Cette catégorie porte sur des informations qui peuvent changer avec le temps ou suivant certains facteurs comme la localisation, temps et date, météo, ainsi que les informations numériques (les comptes des utilisateurs sur les réseaux sociaux, émail, pages Web, documents).

9.3.3 Gestion du contexte

La gestion du contexte est subdivisée en plusieurs volets, l'un porte sur le choix des informations contextuelles les plus pertinentes (soit dépendant de l'application ou du système), le concepteur de l'application est le responsable qui décide quelles informations contextuelles le sont.

Les systèmes doivent être sensibles à leur environnement physique et prendre

en compte la dynamicitè des objets communicants ainsi que l'évolution des paramètres physique de l'environnement, aidant à la prise des décisions les appropriées [M. F. Khalfi, 2014].

10 Conclusion

Ce deuxième chapitre est consacré à l'état de l'art des systèmes de recommandation. Nous avons défini le système de recommandation et son histoire. Nous avons aussi présenté des définitions et des exemples sur les SR basés sur le contenu, filtrage collaboratif et les différentes hybridations existant sont ensuite abordés, avec leurs avantages et inconvénients. Ainsi que les limites des SR comme le problème de démarrage à froid, qui est considéré comme le problème majeur des systèmes de recommandation.

Nous avons ensuite présenté les algorithmes connus dans les SR comme le classificateur naïve bayésienne, forêt aléatoire, Réseau neuronal, K-plus proches voisins (KNN), Régression logistique et Classificateur d'Amplification de gradient. L'évaluation est une phase très importante. Pour cela nous avons présenté les différentes mesures d'évaluation avec leurs formules. Enfin, le chapitre a conclu par la sensibilité aux profils, préférences et contexte des utilisateurs et une conclusion.

Chapitre 3

Travaux connexes

Sommaire

1 Introduction

Nous présentons dans ce chapitre les différents travaux connexes à notre projet. Nous avons choisi les travaux récents qui intéressent et proches de notre axe de recherche, Tels que L'optimisation et les systèmes de recommandation.

L'optimisation a existé depuis longtemps. Elle a connu une évolution remarquable. Nous dressons dans ce chapitre, une étude des articles scientifiques proches de notre proposition, a savoir [J. Arshad, 2020] où les auteurs ont fait une étude comparative entre l'algorithme génétique et les algorithmes d'évolution différentielle, Ainsi dans [J. Nator, 2020], les auteurs ont utilisé l'algorithme génétique dans les contrôleurs du trafic. Dans la littérature nous distinguons plusieurs travaux dans le domaine du contrôle du trafic routier par les algorithmes d'optimisation.

Concernant la partie des systèmes de recommandation, nous avons repéré quelques travaux connexes et d'actualités proches de notre thématique. Ainsi nous avons pu faire une comparaison entre les différentes références. Nous concluons ce chapitre par une conclusion.

2 Analyse de l'existant des travaux relatifs au problème du plus court chemin

Le problème de la détermination des k plus courts chemins a été étudié dans différents travaux. Un chemin plus court optimal est caractérisé par une longueur minimale d'une source à une destination. A cause des problèmes d'applications qui sont nombreux et divers, il y a eu un essor de la recherche sur les algorithmes du plus court chemin. Le calcul de ce dernier peut générer des solutions exactes ou approchées en utilisant des algorithmes exacts ou des algorithmes méta heuristiques [T. Abeywickrama, 2016]. Les algorithmes heuristiques visent à minimiser le temps de calcul et l'espace mémoire [H. Bast, 2015]. De nombreux travaux se concentrent sur les algorithmes qui ciblent les applications du trafic, Dans ce qui suit nous résumons quelques travaux.

Dans [N. Shanmugasundaram, 2019] les auteurs ont appliqué l'AG pour déterminer la conception du chemin de guidage pour un réseau routier à sens unique dans le but de minimiser la distance totale parcourue par les véhicules au sein du réseau. Une technique de branchement et de liaison avec la recherche en largeur d'abord est appliquée pour chercher le chemin le plus court entre les points des sources et les points des destinations.

Dans [J. Arshad, 2020] les auteurs ont fait une comparaison entre l'algorithme génétique et les algorithmes d'évolution différentielle pour améliorer le niveau de service de l'intersection en optimisant le plan de synchronisation des signaux. Les résultats montrons l'efficacité de l'algorithme génétique dans la majorité des expérimentations faites.

Simsir et les autres auteurs [F. Simsir, 2019] ont proposé une solution intelligente pour trouver le chemin le plus court pour le VRPSDP (problème de routage de véhicule avec livraison et ramassage simultanés) à l'aide de l'algorithme de colonie d'abeilles artificielles (ABCA).

Broumi et les autres auteurs [S. Broumi, 2020] ont proposé une approche d'optimisation basée sur une stratégie de classement qui prend en compte à la fois la longueur des arêtes et le nombre de nœuds. Ce travail est très proche de notre proposition.

Nator Junior et les autres auteurs [J. Nator, 2020] visent à concevoir des contrôleurs de trafic actionnés et optimisés par l'algorithme génétique. Aussi ils visent à concevoir des contrôleurs de trafic actionné en utilisant des capteurs de présence, et optimisés par l'algorithme génétique.

Ainsi, il existe d'autre travaux similaires à notre recherche, nous présentons brièvement quelques un, dans [O. Ugurlu, 2018] les auteurs ont présenté un nouvel algorithme génétique hybride. En outre, ils ont proposé un nouvel algorithme heuristique pour MWDS (Minimum Weight Doifying Set) afin de créer une population initiale. En comparant aux algorithmes existants dans la littérature, les résultats expérimentaux montrent que les algorithmes génétiques hybrides sont plus rapides que ces algorithmes, ils peuvent également donner de meilleures solutions.

Dans [S. C. Abraham, 2015], les auteurs tentent de résoudre le problème de trouver un algorithme de chemin le plus court point à point adéquat pour les graphiques de plus grande taille en utilisant à la fois l'algorithme A* et l'algorithme génétique. L'approche bi-directionnelle diminue l'espace de recherche et l'algorithme génétique optimise le problème d'exploration pour renvoyer le meilleur résultat.

Junwoo et les autres auteurs [K. JunWoo, 2018] ont proposé un nouvel algorithme pour calculer les k plus courts chemins simples dans un graphe orienté, et ils ont rendu compte de sa mise en œuvre. Les auteurs prouvent que GA est utilisé pour explorer virtuellement l'espace de recherche.

Dans [D. Goldberg, 1992] les auteurs ont crée un nouveau algorithme pour satisfaire le besoin dans la recherche sur la personnalité, ils ont identifié les k chemins simples les plus courts dans un graphe orienté, et ils ont annoncé sa mise en œuvre.

Sommer [Sommer, 2013] a critiqué les approches, les algorithmes et les résultats choisis sur les requêtes de chemin le plus court à partir de ces champs, en se concentrant principalement sur le compromis entre la taille de l'index et le temps de requête. En outre, l'enquête de Sommer a introduit la classe d'algorithmes du réseau de transport, et il a inclus des algorithmes pour les graphiques généraux ainsi que des graphiques planaires et complexes.

Dans [S. Pettie, 2003] les auteurs ont proposé un nouvel algorithme qui surpasse l'algorithme de Dijkstra sur les graphes clairsemés pour le problème du plus court chemin toutes paires. Les résultats présentés montrent que le nouvel algorithme s'exécute plus rapidement que l'algorithme de Dijkstra sur divers graphes clairsemés lorsque le nombre de sommets varie de quelques milliers à quelques millions.

En 2012, Lee et les autres auteurs [J. Lee, 2012] ont comparé l'algorithme de Dijkstra et l'algorithme génétique et son propre algorithme DGA. Les résultats ont montré que l'algorithme de Dijkstra ne donne pas les meilleures solutions dans le graphique, contenant plus de 10 000 nœuds ; En revanche, l'algorithme DGA a donné des résultats satisfaisantes dans un temps d'exécution acceptable.

En analysant les travaux mentionnés ci-dessus, nous pouvons résumer que :

1. L'algorithme de Dijkstra est considéré parmi les meilleurs algorithmes d'optimisation. Il a été appliqué avec succès à une variété de problèmes d'optimisation. C'est également un algorithme rapide dans le temps d'exécution, cependant, plusieurs recherches ont confirmé que les performances de l'algorithme de Dijkstra ne sont pas optimales dans les graphes de grand tailles [J. Lee, 2012] [Singh, 2018] [Reddy, 2013].

2. Pour résoudre le problème des plus courts chemins, la complexité d'un algorithme exact augmente avec le nombre de sommets du graphe.

3. L'algorithme génétique est très connu et il a une caractéristique biologique. Il fonctionne avec la fonction aléatoire, et c'est l'un des algorithmes heuristiques les plus connus et il est utile dans plusieurs domaines d'optimisation.

Pour ces raisons, un algorithme génétique est suggéré comme la solution optimale au problème des k-plus courts chemins. Le tableau (3.1) montre quelques critères de comparaison de certains travaux connexes d'optimisation tels que : les algorithmes utilisés, l'ensemble de données utilisées et le type d'optimisation.

Référence	Année	Algorithmes	Base de données	Type de graphe
[J. Arshad, 2020]	2020	Algorithme Génétique	Non mentionnée	Non orienté
[F. Simsir, 2019]	2019	Colonie d'abeilles	Benchmark problem datasets	Orienté
[J. Nator, 2020]	2020	Algorithme Génétique	Traffic signal controller	Orienté
[S. Broumi, 2020]	2020	Théorie neutrosophique	Exemple numérique	Non mentionné
[M. Changxi, 2018]	2018	Algorithme Génétique	Généré par l'auteur (aléatoire)	Orienté
[Singh, 2018]	2018	Algorithme de Dijkstra	Carte binaire	Non mentionné
[I. Maatouk, 2017]	2017	Algorithme Génétique	Généré par l'auteur (aléatoire)	Orienté
[Reddy, 2013]	2013	Algorithme de Dijkstra et A*	Non mentionnée	Non orienté
[F. Duchon, 2014]	2014	A*	Planification de trajectoire	Orienté
[O. Ugurlu, 2018]	2018	Algorithme Génétique	deux BDD créées aléatoirement	Les deux

TABLE 3.1 – Travaux connexe d'optimisation

3 Analyse de l'existant des travaux relatifs aux systèmes de recommandation

Avec l'avènement du Web et l'évolution de la technologique Ainsi, l'importante masse de données a utiliser ou a analyser. A tel point qu'il est devenu difficile de savoir quelles données recherchées, ainsi ou les retrouvées. Les techniques informatiques sont développées pour faciliter cette recherche. Ainsi, l'extraction d'informations pertinentes. Il s'agit de guider l'utilisateur dans son exploration des données afin qu'il trouve les informations pertinentes. De nombreux travaux modernes se concentrent sur les systèmes de recommandation, dont nous résumons dans ce qui suit.

Jigarkumar et al. [H. S. Jigarkumar, 2020] ont crée un nouveau système de recommandation de voyage hybride sous titre « Hybrid User Clustering-based Travel Planning System for Personalized Pointof-Interest Recommendation ». Le système combine entre le filtrage collaboratif et le clustering, pour prédire des éléments très similaires à l'utilisateur cible actif en temps réel. Les auteurs ont appliqué leur système ainsi que d'autres systèmes connues déjà sur les deux bases de données Yelp et TripAdvisor. Leurs résultats montrent l'efficacité de l'algorithme hybride par rapport aux autres systèmes de recommandations comme : LBPARS, LFARS et d'autres.

Dans [K. L. Umesh, 2021], les auteurs ont créé un système de recommandation récent et efficace intitulé « Méthode de forêts aléatoires pondérées hybrides pour la prédiction et la classification des clients acheteurs en ligne ». Les auteurs ont utilisé la méthode de classification de forêt aléatoire qui appartient à l'apprentissage automatique. Ils ont utilisé un data-set, extrait de Kaggle, qui contient 80000 consommateurs d'un site de vente en ligne, donc ils ont prédit si un acheteur a acheté un article donné ou non. Les métriques suivantes sont utilisé pour évaluer leur système : Rappel, Précision, F-mesure, Accuracy.

Khorsand et al. [R. Khorsand, 2020] ont fait en 2020 une comparaison entre 8 modèles d'apprentissage automatiques supervisés, ils ont récupérés tous les avis

sur tous les hôtels de Téhéran sur TripAdvisor, ensuite, ils ont appliqués les 8 modèles suivants sur le data-set : K-plus proches voisins (KNN), Naïve Bayes, Arbre de décision, Régression logistique, Vecteur de support machine, Réseau de neurones, Forêt aléatoire, Amplification de gradient. Les auteurs ont utilisé 70 % de ces données pour l'apprentissage et 30 % pour le test, et ils ont utilisés plusieurs métriques pour faire les comparaisons, nous citons : Accuracy, MAE, K-S Test, Tukey Test. Il est a noté que dans ce papier les auteurs ont fait une très bonne partie de traitement de données, et le meilleur algorithme pour s'adapter à l'ensemble de données des hôtels de Téhéran est l'algorithme K-plus proche voisins (KNN).

D'après l'auteur de l'article [V. K. Muneer, 2020], la pacification d'un voyage est une tâche chrono-phage et herculéenne pour les voyageurs inexpérimentés. Et ce point représente la problématique de ce papier, ils ont procédé à une revue détaillée et des évolutions chronologiques de diverses méthodes et techniques utilisées dans le secteur du voyage et du tourisme, ensuite, ils ont comparé leur efficacité dans les recommandations. Ils ont classifiés les systèmes de recommandation en trois classes (filtrage collaboratif, technique basée sur le contenu, filtrage hybride), sachant que la technique basée sur le contenu contienne deux sous classes (filtrage basé sur un modèle et filtrage basé sur un mémoire). La figure (3.1) représente le schéma des classes des systèmes de recommandations.

Les auteurs ont divisés les points d'intérêts en cinq domaines (hôtel, restaurant, tourisme divers, attraction et autre), la figure (3.2) représente les catégories du tourisme utilisées dans le papier.

Comme résultat, les auteurs ont classifié 17 travaux connexes par ordre selon l'importance et les résultats obtenus par les systèmes de recommandation. Ce Survey est très intéressant car il donne une analyse générale sur les différents SR existants dans le domaine du tourisme [V. K. Muneer, 2020].

En 2017, Vellaichamy et Kalimuthu [V. Vellaichamy, 2017] ont proposé un système de recommandation de film basé sur le filtrage collaboratif hybride qui combine le clustering Fuzzy C Means (FCM) avec l'optimisation Bat afin de réduire

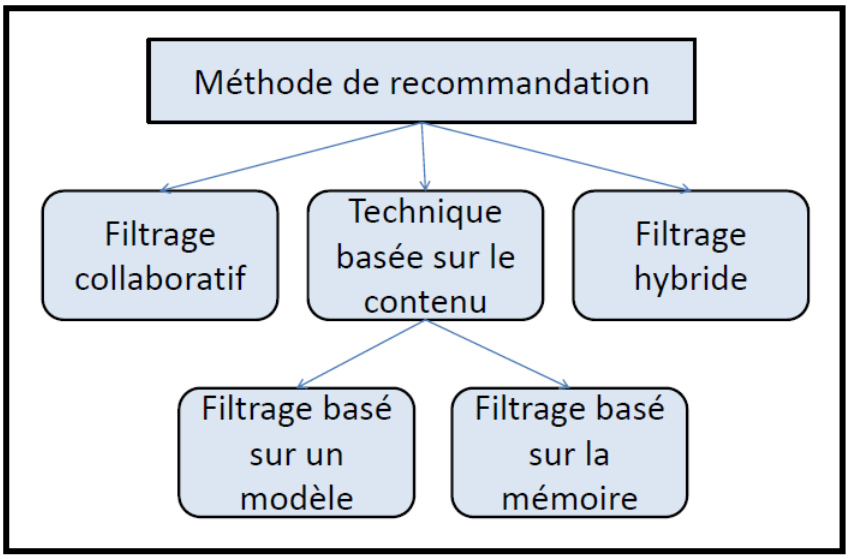


FIGURE 3.1 – Schéma des classes des systèmes de recommandation

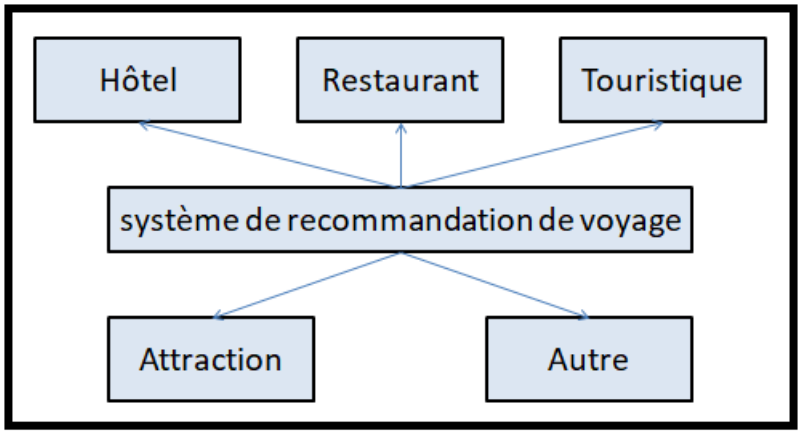


FIGURE 3.2 – Catégories des points d'intérêts utilisées dans l'article [V. K. Muneer, 2020]

le problème d'évolutivité et améliorer le clustering, ce dernier est utilisé pour regrouper les utilisateurs en différents groupes. L'algorithme Bat est utilisé pour obtenir la position initiale des clusters. Le système proposé a été évalué sur l'ensemble de données Movie Lens. Les résultats des expériences obtenus montrent que l'algorithme proposé pourra donner de meilleurs résultats de recommandation par rapport à d'autres techniques de clustering comme PCA–GAKM, PCA-KM, K-Means, UPCC (Pearson correlation coefficient based method) et SOM en termes d'erreur absolue moyenne (MAE), de précision et de rappel.

les auteurs dans [W. Shafqat, 2020] proposent la recommandation de sensibilisation au contexte de localisation du système en utilisant le modèle hiérarchique, qui est basé sur mémoire à terme (LSTM) (Long-Short Term Memory). Ce modèle prédit la probabilité de la prochaine visite d'un utilisateur dans un site touristique et intègre ensuite des données contextuelles pour recommander les meilleurs endroits à l'utilisateur. Le LSTM proposé fonctionne mieux sur le jeu de données que le Naïve Bayes. Les deux algorithmes sont testés sur une base de données crée par les utilisateurs, et il ont utilisé l'accuracy comme technique d'évaluation.

Ce travail [C.S. Namahoot, 2015] propose un système de recommandation sensible au contexte nommé CAT-TOURS qui utilise une ontologie et l'algorithme NB. Il permet de trouver une simple méthode pour catégoriser les documents web du tourisme en Thaïlande contenant des informations sur plus d'un sujet et prendre du temps compte des contraintes dans la formulation des recommandations. Cet ouvrage recommandait des informations touristiques en fonction du temps et la saison en appliquant l'ontologie temporelle. ils ont utilisés la précision, rappel et f-mesure comme techniques d'évaluations.

NTRS [T. Ucar, 2019] est un nouveau système de recommandation de voyage contextuel développé pour générer des alternatives destinations de voyage pour les clients. L'approche proposée est basé sur des méthodes hybrides de fouille de données en combinant classification (ANFIS, RBFN et Naîve Bayes) et algorithmes de clustering (X-means et Fuzzy C-means). Dans cette étude, le but

principal est de trouver le meilleur modèle de prédiction pour fournir des trajets alternatifs via NTRS.

Le tableau (3.2) montre quelques critères de comparaison de certains travaux connexes en recommandation tels que : les techniques utilisés, l'ensemble de données et les métriques d'évaluations utilisés.

Référence	Année	Technique	Domaine	Base de données	Métrique d'évaluation
[H. S. Jigarkumar, 2020]	2020	Hybride	Voyages	Yelp et TripAdvisor	Rappel, Précision, F-mesure, Accuracy
[K. L. Umesh, 2021]	2021	Forêt aléatoire	Achats	Kaggle	Rappel, Précision, F-mesure, Accuracy
[R. Khorsand, 2020]	2020	KNN	Hôtels	TripAdvisor	Accuracy, MAE, K-S Test, Tukey Test
[V. K. Muneer, 2020]	2020	Survey	Voyages	Non mentionné	Accuracy
[V. Vellaichamy, 2017]	2017	Hybride	Films	Movie Len	MAE, Précision, Rappel
[W. Shafqat, 2020]	2020	ML	Voyages	Crée par l'auteur	Accuracy
[C.S. Namahoot, 2015]	2015	NB	Voyages	ET en Thaïlande	Rappel, Précision, F-mesure
[T. Ucar, 2019]	2019	Hybride	Voyages	Vols et Hôtels	RMSE, TPR, TNR, Précision

TABLE 3.2 – Travaux connexes de la recommandation

4 Divers domaines des systèmes de recommandation

La recommandation est utilisée dans des divers domaines, et elle a facilité le choix malgré l'existence des différentes goûts et préférences, nous présentons dans les prochaines lignes quelques domaines de recommandation :
1/ Culture : le domaine le plus connue puisque on trouve la recommandation dans les différents sites et applications culturelle (films ,musiques, musées, etc.).
2/ Tourisme : c'est un domaine intéressant, et il à marqué un très bon évolution dans les dernières années dans plusieurs points d'intérêts comme les restaurants, hôtels, attraction, voyage organisé, billet d'avion, etc.
3/ Commerce : la publicité est remplacé par la recommandation puisque cette dernière est efficace, et elle propose le bon produit dans la bonne période avec une intelligence et un précisément énorme.
4/ Industrielle : c'est un nouveau domaine ou la recommandation aide les machines industrielles à économisé l'énergie et pour amélioré la qualité du produit d'après les avis des clients.

5 Conclusion

Dans ce chapitre, Nous avons présenté quelques travaux connexes afin de faire une synthèse générale et de les comparer avec notre proposition. Deux analyses de l'existant ont été établies, une sur le concept du plus court chemin et la deuxième concerne la recommandation. Nous avons aussi résumé dans un tableau comparatif une analyse et une comparaison entre les travaux connexes selon plusieurs critères à savoir : l'année de création, les algorithmes, les techniques et la base de données utilisées. Nous avons aussi présenté les divers domaines des systèmes de recommandation et Nous avons terminé ce chapitre par une conclusion.

Deuxième partie

Contributions

Chapitre 4

Approche hybride d'optimisation du problème du plus court chemin

Sommaire

1 Introduction

Le chapitre 3 a mis en évidence les avantages et les limitations des approches actuelles dédiées au problème d'optimisation du plus court chemin et la recommandation de services. Nous avons identifié également quelques besoins et défis à surmonter afin d'améliorer le processus de la recommandation. Dans ce chapitre, nous proposons un algorithme hybride très efficace, nommé IOGA [M. Y. Hayi, 2022], répondant à un ensemble de besoins et défis. Nous illustrons également les motivations, les principes et les mécanismes de base de notre contribution et un ensemble de résultats expérimentaux.

2 Motivation et contribution

Le nombre de véhicules sur la route a augmenté de façon extraordinaire. Par conséquent, la gestion du trafic est devenue un problème majeur. Dans cette partie, nous allons étudié le réseau de trafic qui concerne les problèmes de routage à grande échelle en tant que domaine d'application. Plutôt que d'utiliser certains algorithmes standards, il est préférable d'utiliser les algorithmes heuristiques des plus courts chemins car ils sont optimaux et significatifs dans toutes les applications de transport.

La méta-heuristique que nous proposons consiste à un nouvel algorithme génétique hybride nommé IOGA (Improved Optimization Genetic Algorithm) qui combine un algorithme approché *(algorithme génétique)* avec un algorithme exact *(algorithme de Dijkstra)* afin de calculer les k-plus courts chemins.

C'est une nouvelle approche hybride basée sur la méta-heuristique et aussi c'est une technique d'optimisation pour réduire le temps d'exécution dans l'algorithme des plus courts chemins. L'algorithme proposé utilise une stratégie qui s'est avérée efficace pour résoudre le problème du plus court chemin. Le problème consiste à trouver un ou plusieurs k chemins optimaux qui minimisent les métriques *distance* et *temps*, du nœud d'origine au nœud de destination dans un graphe orienté ou non orienté. Les tests sont effectués à l'aide de l'algorithme Dijkstra et d'algorithmes méta-heuristiques (algorithme génétique, algorithme

A*) sur des instances de réseau routier gérées de manière aléatoire. L'étude expérimentale démontre l'efficacité de notre approche proposée en termes de temps d'exécution et de qualité des résultats. Les résultats empiriques obtenus montrent que l'IOGA est meilleur que certaines techniques existantes en terme de solution optimale à chaque génération.

Dans ce qui suit, nous allons présenter en détail l'approche proposée à savoir la création de deux bases de données, ainsi que les bibliothèques utilisées par python pour appliquer les différents algorithmes sur nos data-sets. De même l'algorithme IOGA, qui est un algorithme d'optimisation utilisé pour la gestion intelligente du trafic routier sur de grands graphiques pondérés. Nous terminons ce chapitre par une discussion des résultats obtenus.

3 Approche proposée

La recherche en méta-heuristique est bien connue dans le domaine de l'intelligence artificielle et a également été très active au cours de la dernière décennie. Face à de nouveaux problèmes d'optimisation complexes, il est tout à fait naturel d'utiliser des techniques de méta-heuristiques. Avec la croissance massive de l'information, l'optimisation du Big Data est devenue l'une des principales recherches. De plus, des réseaux complexes tels que les réseaux de transport, de communication, Internet et de distribution, etc., ont de plus en plus attiré l'attention de divers domaines de la science et de l'ingénierie. La combinaison de la méta-heuristique avec d'autres techniques d'optimisation, appelées méta-heuristiques hybrides, peut offrir une flexibilité avancée et un comportement plus efficace lorsqu'il s'agit de résoudre des problèmes à grande échelle et du monde réel.

3.1 Bases de données

Un graphe est un réseau complexe important qui permet de décrire la relation entre un ensemble de nœuds inter-connectés (entités). Le calcul du plus court chemin est une opération fréquente. Dans le cas des problèmes *complexes*, il est

souvent intéressant de pouvoir trouver les k plus courts chemins *optimaux*. Au cours de la dernière décennie, le problème des chemins les plus courts sur des graphes complexes est une tâche difficile ; il est également considéré comme l'un des problèmes d'optimisation combinatoire les plus étudiés. Beaucoup d'algorithmes de graphes déjà existants ne sont pas appropriés pour les graphes de taille importante [J. Lee, 2012].

Nous avons généré aléatoirement par python deux réseaux de trafic routier orientés et non orientés. Nous avons appliqué notre algorithme sur deux jeux de données, le premier contient 10 000 nœuds, et il s'agit d'un jeu de données orienté, et le deuxième est un jeu de données non orienté qui contient 5 000 nœuds, chaque nœud représente une ville. La génération des deux jeux de données a été faite selon les travaux de [J. Lee, 2012], nous avons généré les deux jeux de données aléatoirement et sauvegardés en plusieurs formats, la distance entre deux nœuds est confinée entre 0 et 9999 (peut ne pas avoir de relation entre deux nœuds), cela signifie que ce n'est pas un graphe complet, et aussi nous avons généré aléatoirement pour chaque graphe 500 paires source/destination comme il est expliqué dans le tableau (4.1).

Numéro de Couple	Source	Destination
Couple 1	N 4835	N 1082
Couple 2	N 619	N 3690
...	...	...
Couple 500	N 2747	N 1839

TABLE 4.1 – Matrice des couples Source / Destination

La figure (4.1) représente les graphiques des deux bases de données en trois dimensions.

Dans notre travail, nous étudions le réseau de trafic dans les problèmes de routage à grande échelle comme domaine d'application. Trouver un itinéraire approprié pour atteindre la destination à temps très réduit avec une économie d'énergie a une importance dans les applications du monde réel [F. Simsir, 2019]. Les sommets (nœuds) du graphe correspondent aux unités spatiales (villes) du

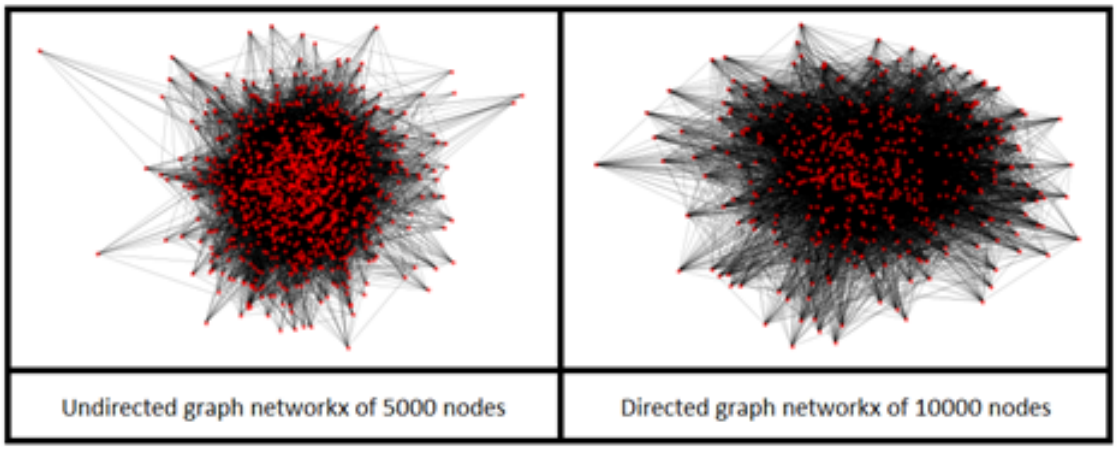


FIGURE 4.1 – Graphes des deux data-sets

réseau routier. Les arêtes relient les sommets et elles sont pondérées avec des poids non négatifs en fonction de la distance entre les sommets [J. Lee, 2012]. Les opérations sur un graphe peuvent prendre beaucoup de temps en raison de la complexité de la connectivité structurelle et de la taille du graphe. De plus, un graphe complexe rend l'efficacité encore plus difficile [Hosseinabadi, 2018].

Afin de trouver le chemin le plus court à partir de la source jusqu'à la destination où les distances entre chaque deux villes sont connues, nous avons créé deux bases de données. La figure (4.2) représente un capture d'écran sur une partie de la base de donnée non orienté crée.

	A	B	C	D	E	F	G
1		**N 1**	**N 2**	**N 3**	**N 4**	**N 5**	**N 6**
2	**N 1**	0	630	1176	1415	2713	865
3	**N 2**	630	0	895	2797	791	1727
4	**N 3**	1176	895	0	1504	653	482
5	**N 4**	1415	2797	1504	0	426	854
6	**N 5**	2713	791	653	426	0	335
7	**N 6**	865	1727	482	854	335	0

FIGURE 4.2 – Capture de la BDD non orientée

L'algorithme 6 présente le code qui fait la création du data-set non orienté pour un nombre de noeuds donnée comme entré.

Algorithm 6 Data-sets Creation

Input Length
Output AdjacentMatrix
Begin
$i \leftarrow 0$
while $i < Length$ **do**
  $j \leftarrow 0$
  while $j < i$ **do**
    $AdjacentMatrix[i][j] \leftarrow random(0, 9999)$
    $AdjacentMatrix[j][i] \leftarrow AdjacentMatrix[i][j]$
    $j \leftarrow j + 1$
  end while
  $AdjacentMatrix[i][i] \leftarrow 0$
  $i \leftarrow i + 1$
end while
Return (AdjacentMatrix)
End

3.2 L'algorithme hybride proposé IOGA

Diverses applications telles que les applications de transport et de recommandation nécessitent l'utilisation d'un chemin plus court pour une meilleure optimisation.

Comme première contribution, nous proposons une méta-heuristique hybride pour résoudre le problème de routage dans les réseaux routiers qui sera par la suite utilisé dans notre deuxième approche de recommandation proposée.

Les algorithmes optimaux du chemin le plus court tels que Dijkstra ont tendance à être extrêmement gourmands en calculs pour les applications en temps réel dans les réseaux de trafic pratiques [Singh, 2018]. Pour cette raison, nous développons un nouvel algorithme combinant l'algorithme exact (à savoir l'algorithme de Dijkstra) avec un algorithme méta-heuristique (à savoir l'algorithme génétique - GA) dans un processus de recherche du chemin le plus court. Dans cette étude, le nouvel algorithme hybride appelé IOGA (Improved Optimization Genetic Algorithm), est développé pour trouver un ou plusieurs k chemins optimaux ayant le coût minimal entre deux sommets (source/destination) dans un graphe orienté et un autre non orienté. Ainsi, l'hybridation d'un algorithme

génétique avec l'algorithme de Dijkstra est une idée intéressante car la stratégie classique utilise l'algorithme de Dijkstra pour minimiser l'espace de recherche alors que la stratégie heuristique optimise le problème de recherche pour renvoyer les meilleures solutions. La structure d'IOGA est résumée dans l'algorithme 7.

L'algorithme d'IOGA contient six entrés, nous les présentent dans les prochaines lignes :

1/ Le graphe (matrice d'adjacente) : est une matrice qui contient les relations entre les noeuds.

2/ Top matrix : est une matrice qui contient les TOP noeuds proches de chaque noeud du graphe, cette matrice est bien expliqué dans la partie (3.3 Top matrix).

3/ TOP : est une variable d'entrer dans l'algorithme de IOGA, elle représente le nombre des nœuds proches d'un nœud quelconque. C'est-à dire pour chaque nœud existant dans la population on aura TOP proches nœuds.

4/ SIZE : est une variable d'entrer dans l'algorithme d'IOGA, elle représente le nombre maximal des nœuds d'une route entre la source et la destination, c'est à dire une solution de SIZE = 6 représente un nœud de source et un nœud de destination et 4 nœuds intermédiaires entre la source et la destination. Dès quand aura un SIZE élevé, le temps d'exécution augmente et on obtient la chance d'avoir la solution optimale. et le contraire si on a un SIZE de petite valeur (par exemple 2 ou 3), on aura un temps d'exécution très faible mais on n'a pas une bonne chance d'avoir la solution optimale.

5/ Source

6/ Destination

Avec l'algorithme Dijkstra, nous calculons le nœud le plus proche allant d'un nœud à un autre. Par contre dans IOGA, nous calculons les nœuds TOP les plus proches (2, 3, 4, etc.), pour avoir plusieurs routes proposées, nous les mettons par la suite dans la population que nous construisons avec IOGA. Après chaque itération, nous calculons la distance de chaque route existante dans la population, nous gardons que les lignes qui peuvent être une solution optimale, et nous supprimons les autres lignes, de sorte que nous n'aurons pas une seule route,

Algorithm 7 Improved Optimization Genetic Algorithm

```
1.  Input
2.    Graph (Adjacency matrix)
3.    Top matrix
4.    TOP
5.    SIZE
6.    Source
7.    Destination
8.  Output
9.    The best solution find
10.    A list thant contains one or more routes that match the solution found
11.  Begin
12.  Add the source to the first population cell
13.  i ← 2
14.  While  (i<SIZE) do
15.    j ← 1
16.    While (j<=TOP) do
17.        Add the (j) node closest to the previous element in population
18.        j ← j+1
19.    End while
20.    k ← 0
21.    While (k<length(population)) do
22.        solution ← FinalFunction(population, destination)
23.        k ← k+1
24.    End while
25.    i ← 0
26.    While (i<length(population)) do
27.        population ← cleaning(population, solution)
28.        j ← j+1
29.    End while
30.    i ← i+1
31.  End while
32.  Return  (population, solution)
33.  End
```

mais nous obtiendrons plusieurs voies de la solution optimale de manière rapide.

Étant donné un graphe G ayant un ensemble de sommets reliées par des arêtes avec des poids d'arête non négatifs. Soient deux sommets s (source) et t (destination), le problème consiste à trouver les k plus courts chemins de s à t. Un entier positif k (nombre de chemins) est renvoyé à la fin de l'itération de l'IOGA.

IOGA trouve de nombreuses solutions, à partir de la source, il lui ajoute les nœuds TOP (c'est une variable) les plus proches de cette source, de sorte qu'il constitue une population de taille TOP^{SIZE-2}.

La figure (4.3) représente un exemple explicatif du fonctionnement d'IOGA.

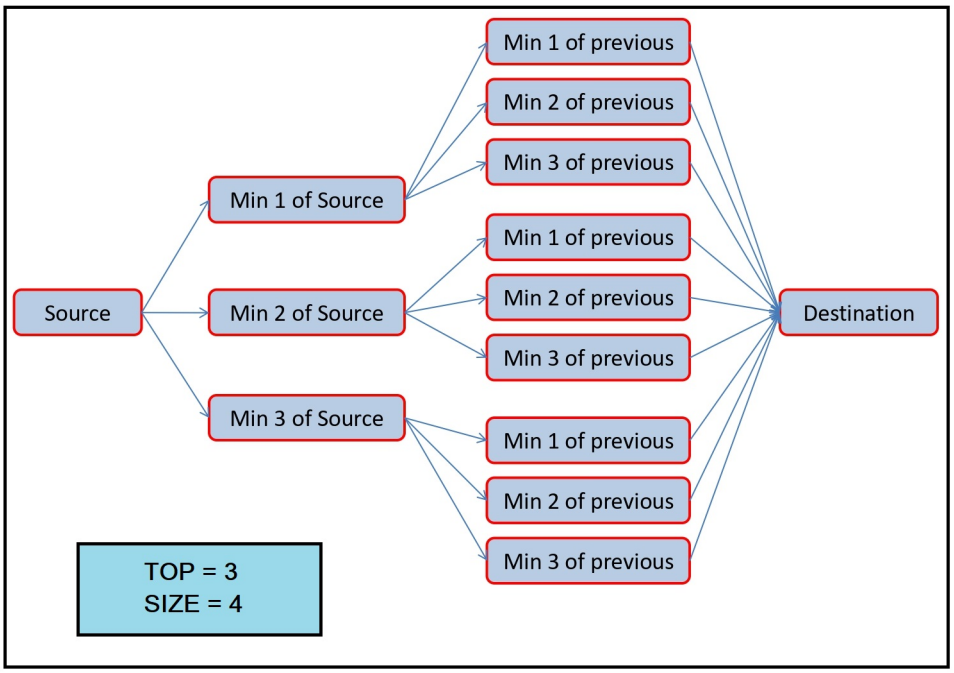


FIGURE 4.3 – Exemple explicatif du fonctionnement d'IOGA

3.3 Top matrix

Top matrix est une matrice utilisée comme entrée dans IOGA, cette matrice contient N lignes et TOP colonnes, chaque ligne représente un nœud et chaque colonne représente le nœud le plus proche. Soit TOP nœuds les plus proches, si

nous prenons par exemple la ligne 10 et la colonne 5, nous trouverons le cinquième nœud le plus proche du nœud numéro 10.

Nous pouvons trouver des cases vides car la variable TOP peut être plus grande que le nombre des voisins d'un nœud donné, par exemple nous pouvons avoir une variable TOP = 5, mais le nœud a seulement 3 arcs sortants, dans ce cas les deux dernières cases restent vides.

IOGA contient quelques fonctions que nous expliquons dans les lignes suivantes.

3.4 Population

C'est une matrice qui contient plusieurs routes, en général, dans un GA, cette matrice est générée aléatoirement, par contre dans IOGA, elle est générée suivant des règles, par exemple, si on prend le paramètre TOP = 6, à partir de la source, nous avons 6 premiers chemins proposés pour la population, les 6 chemins commençant par les 6 nœuds les plus proches de la source, et ainsi de suite. Nous aurons une population qui a une taille dynamique, car, après chaque itération (SIZE) on supprime les chemins qui ne permettent pas de donner des solutions optimales en utilisant la fonction Nettoyage.

3.5 Fonction de nettoyage

Le but de la fonction de nettoyage est d'éviter les solutions irréalisables ou non optimaux et précisément les routes ayant une longueur plus grande que la solution déjà trouvée. A la fin de chaque itération (SIZE), nous appliquons la fonction de nettoyage sur la population afin de supprimer les solutions qui ne peuvent jamais être inférieures à la solution MIN que nous avons trouvée jusqu'à présent.

La fonction Nettoyage est appliquée à la fin de chaque partie de SIZE, par exemple si le paramètre SIZE = 8, nous appliquons la fonction Nettoyage 7 fois, (nous n'appliquons pas cette fonction dans la première phase qui contient seulement la source). La fonction Nettoyage a deux objectifs :

• Fitness : c'est une fonction objective qui calcule la solution de chaque solu-

tion de la population à un instant donné, et nous comparons cette solution avec la meilleure solution trouvée.

• Nettoyage : c'est pour supprimer chaque ligne de la matrice qui existe dans la population et qui donne un résultat fitness plus grand que la meilleure solution trouvée pour l'instant. En général, la fonction Nettoyage permet de calculer les solutions des différents itinéraires qui existent dans la population, et supprimer chaque itinéraire qui ne peut jamais être une solution optimale, afin de réduire la taille de la population et donc gagner du temps.

L'algorithme 8 représente la fonction de Nettoyage.

Algorithm 8 Cleaning function

```
Input
    population
    solution
Output
    population
Begin
i ← 0
While (i<length(population)) do
    j ← 1
    s ← 0
    While (j<length(population[i])) do
        s ← s+distance(population[i][j-1],population[i][j])
        j ← j+1
    End while
    If (s>solution)then
        Del (population[i])
    End if
    i ← i+1
End while
Return  (population)
End
```

3.6 Fonction finale

Nous vérifions par la fonction finale s'il existe un chemin direct entre le dernier élément de la ligne de la population avec la destination à la fin de chaque itération, si c'est le cas nous disons que la ligne représente une solution réalisable, et nous comparons cette solution réalisable avec notre meilleure solution, et bien sûr si la solution faisable est inférieure à la meilleure solution, nous gardons la solution faisable comme meilleure solution, et aussi nous sauvegardons cette ligne de population car elle représente le chemin de la meilleure solution trouvée dans un moment donné. Et si la nouvelle solution égale à la meilleure solution, nous sauvegardons le chemin de cette solution, car notre algorithme récupère tous les meilleurs chemins qui existent. L'algorithme 9 représente la fonction finale.

4 Expérimentation

Afin de vérifier les performances de notre approche, nous avons implémentés par python l'algorithme proposé IOGA, l'algorithme de Dijkstra, l'algorithme A* et l'algorithme génétique, et toutes les expériences sont réalisées sur Intel (R) Xeon (R) cpu E5 - 2620 v3 2.40 ghz 2,40 GHz (16 Go de RAM, processeur 64 bits OS x64). Nous avons généré aléatoirement deux réseaux de trafic routier orientés et non orientés que nous avons expliqué dans la partie (3.1).

5 Exemple numérique

L'algorithme que nous avons créé est très compliqué, donc pour mieux comprendre le déroulement de l'algorithme nous présentons ici un exemple simple pour comprendre les différentes étapes de notre proposition :

• Le paramètre TOP = 3, c'est-à-dire qu'à partir de chaque nœud, on prendra les nœuds TOP (3) les plus proches.

Algorithm 9 Final function

```
Input
    population
    destination
    solution
Output
    solution
Begin
i ← 0
While (i<length(population)) do
    var ← length(population[i])-1
    If (exists-direct-path(population[i][var],destination)) then
        s ← 0
    End if
End while
j ← 0
While (j<var) do
    s ← s + distance(population[i][j],population[i][j+1])
    j ← j + 1
End while
s ← s + distance(population[i][j],destination)
If (s<solution) then
    solution ← s
End if
i ← i + 1
Return (solution)
End
```

• Le paramètre SIZE = 4, c'est-à-dire que dans chaque ligne de la population nous avons une source, deux nœuds et une destination.

Nous avons proposé un graphe de 10 nœuds pour appliquer l'exemple. Le tableau (4.2) représente la matrice d'adjacence qui représente le graphe de l'exemple.

	N1	N2	N3	N4	N5	N6	N7	N8	N9	N10
N1	0	15		6	23		19	23		
N2	15	0				16		29		8
N3			0		13			15		
N4	6			0		42	16		26	17
N5	23		13		0	53		19		
N6		16		42	53	0			11	16
N7	19			16			0	5	82	
N8	23	29	15		19		5	0		
N9				26		11	82		0	5
N10		8		17		16			5	0

TABLE 4.2 – Matrice d'adjacence qui représente le graphe de l'exemple

Pour cela nous proposons un exemple qui explique le déroulement de l'algorithme, nous choisissons aléatoirement une source (N7) et une destination (N9). Nous initialisons le paramètre TOP = 3 et SIZE = 4, puis nous lançons l'algorithme en fonction de la variable SIZE (2, 3, 4).

• SIZE = 2

Nous vérifions s'il existe une route directe de la source à la destination. Donc pour cet exemple il y a un chemin direct entre N 7 et N 9 avec une distance égale à 82. La figure (4.4) (SIZE = 2) représente un schéma explicatif de cette étape d'où la source est en vert et la destination est en rouge.

• SIZE = 3

La figure (4.4) illustre l'explication de la partie de SIZE = 3 : les trois boules noires représentent les 3 nœuds (N8, N4, N1) les plus proches de la source, après cela nous vérifions s'il y a une relation directe entre les 3 nœuds (N8, N4, N1) et la destination. Après vérification, nous avons trouvé une relation uniquement entre le nœud 4 et la destination par une valeur = 26, et il n'existe pas de relation directe entre les deux autres nœuds (N8, N1) et la destination. Nous avons donc qu'une seule solution réalisable (source, N4, destination) avec une distance de 42. Cette dernière est inférieure à la précédente (82), donc nous gardons 42 comme meilleure solution jusqu'à maintenant.

• SIZE = 4

Pour la partie de (SIZE = 4), nous avons 3 nœuds les plus proches de la source et qui représentent la partie précédente, et à partir de chaque nœud, nous aurons également 3 nœuds, nous aurons donc 9 routes de la source à la destination. Nous pouvons avoir une ou plusieurs routes bloquées car parfois nous n'aurons pas une relation entre l'un des nœuds et la destination. Par conséquent, cette route sera ignorée. Cette partie est bien expliquée dans la figure (4.4) (SIZE = 4).

6 Résultats et discussion

L'étude expérimentale compare notre méthode hybride approximative (i.e., IOGA) avec trois algorithmes (algorithme de Dijkstra, algorithme génétique et algorithme A*) afin d'assurer les contributions spécifiques suivantes :

i) Trouver un ou plusieurs chemins les plus courts.

ii) Minimiser le temps d'exécution en comparant avec les trois algorithmes précédemment mentionnés.

L'efficacité de notre approche utilisant l'algorithme heuristique est confirmée.

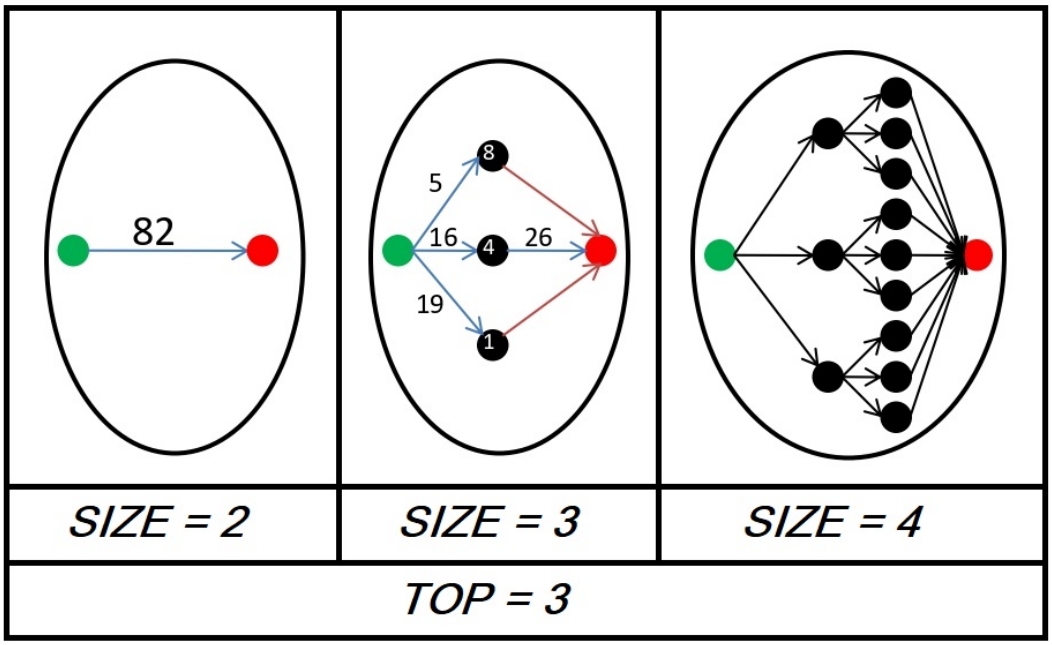


FIGURE 4.4 – Étapes de l'algorithme IOGA pour SIZE = 2, 3 et 4

Les résultats finaux illustrent que l'approche proposée avec la nouvelle stratégie d'optimisation nous a permis d'obtenir de bons résultats avec des performances élevées.

Nous présentons dans les prochaines lignes nos résultats obtenus ainsi que la discussion de chaque résultat. Nous avons divisé cette partie d'après le type de la base de données utilisé en : graphe orienté et graphe non orienté.

6.1 Graphe orienté

Pour le graphe orienté, nous allons expliquer les résultats des 10 couples que nous avons choisis au hasard, pour voir en détail les résultats obtenus. Le tableau (4.3) illustre le nombre des solutions et le temps d'exécution en secondes de chaque algorithme pour 10 paires source/destination par l'utilisation des paramètres (TOP = 6 et SIZE = 10).

Algorithmes	Dijkstra		IOGA	
N. Couple	Nombre de solutions	TE (s)	Nombre de solutions	TE (s)
Couple 375	1	2.2	5	4.3
Couple 163	1	2.6	4	4.1
Couple 29	1	1.8	4	3.8
Couple 407	1	1.6	3	4.2
Couple 149	1	1.8	3	2.9
Couple 380	1	2.3	2	1.7
Couple 293	1	1.7	1	2.3
Couple 73	1	2	4	1.7
Couple 418	1	1.9	1	2.7
Couple 205	1	2.4	3	1.9
AVG	1	1.98	2.66	3.07

TABLE 4.3 – Résultats du graphe orienté - nombre de solutions

Nous avons remarqué que l'algorithme de Dijkstra donne toujours une solution unique, que l'on nomme une seule route, par contre IOGA donne entre une et 5 solutions, et avec une moyenne de 2,66 solutions pour chaque couple (AVG de 500 paires source/destination). Pour le temps d'exécution, l'algorithme de Dijkstra a une moyenne de 1,98 seconde pour les 500 couples, et IOGA a marqué 3,07 secondes, mais il récupère plusieurs itinéraires pour chaque couple. Donc par moyenne, IOGA prend 1.15 seconde pour chaque solution et Dijkstra prend 1.98 seconde.

Nous avons comparé le temps d'exécution de chaque solution pour les 10 couples. Par exemple, le couple 375 qui a obtenu par l'algorithme IOGA 5 solutions en 4,3 secondes, soit une moyenne de 0,86 seconde pour chaque solution, avec l'algorithme de Dijkstra, uniquement une solution en 2,2 secondes. Par conséquent, nous remarquons que IOGA retourne plusieurs solutions en une moyenne de temps plus faible que le temps nécessaire que Dijkstra requiert pour récupérer une solution.

La figure (4.5) représente un graphique de comparaison du temps d'exécution entre les deux algorithmes Dijkstra et IOGA par l'utilisation des paramètres (TOP = 10 et SIZE = 12).

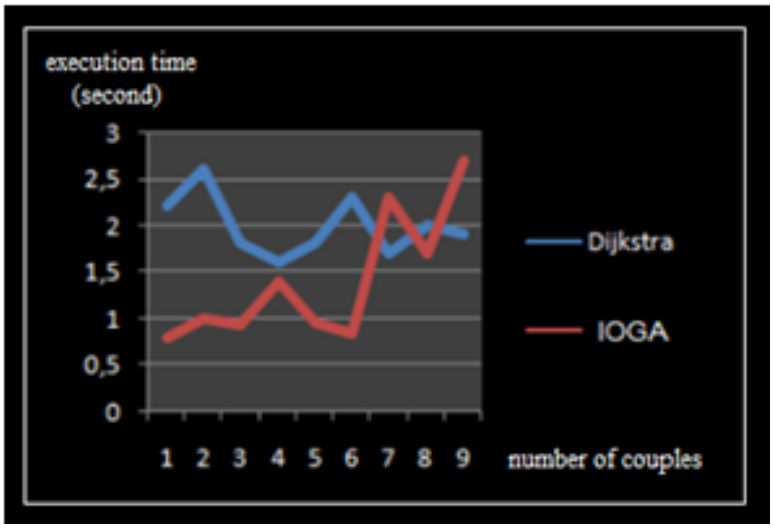


FIGURE 4.5 – Comparaison du temps d'exécution entre IOGA et Dijkstra

La figure (4.6) représente le graphe des nombres des chemins obtenus par IOGA pour les 100 premiers couples (source/destination) avec les paramètres (TOP = 8 et SIZE = 10). On note que IOGA n'a pas trouvé 4 solutions sur 100 par contribution à l'algorithme de Dijkstra (73, 82, 93, 96), en revanche, IOGA a trouvé les mêmes solutions que l'algorithme de Dijkstra dans 62 cas (une seule solution), mais IOGA récupère dans 34 cas plusieurs solutions (de 2 à 8 chemins) sachant que l'algorithme de Dijkstra renvoie toujours une seule route. La meilleure solution est marquée pour le couple numéro 87, donc IOGA renvoie 8 chemins différents de la même solution que l'algorithme de Dijkstra, et nous pouvons avoir d'autres solutions si nous exécutons le programme avec des valeurs plus élevées des paramètres TOP et SIZE. Nous avons également présenté nos résultats dans le tableau (4.4) qui contient le nombre des chemins pour les 100 premières paires (source/destination) par apport aux paramètres de IOGA (TOP et SIZE).

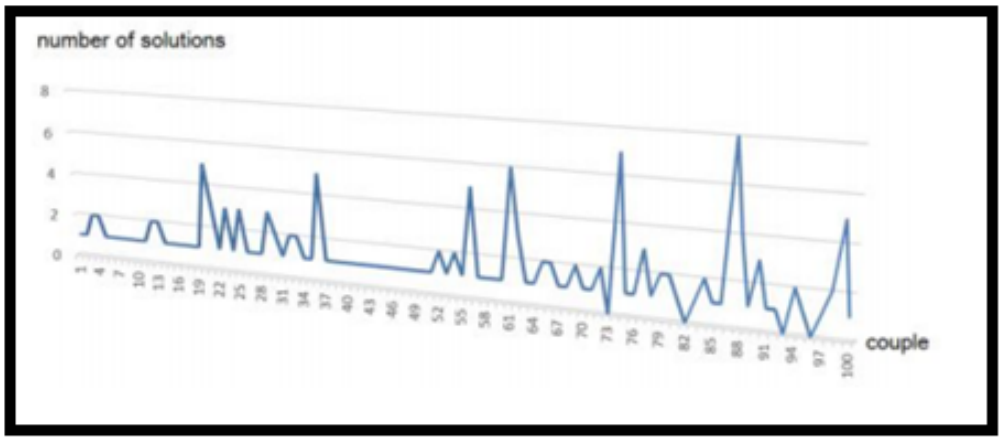


FIGURE 4.6 – Représentation graphique du nombre de solutions pour les 100 premiers couples

Pour le paramètre (TOP = 10 et SIZE = 10), nous avons eu un cas où notre algorithme ne donne pas le même résultat que l'algorithme de Dijkstra (un résultat inférieur à celui de l'algorithme de Dijkstra). Nous avons également eu 64 cas sur les 100 cas où IOGA ne devait donner qu'un seul chemin de la même solution que l'algorithme de Dijkstra et le plus important. IOGA renvoie deux solutions dans 15 cas sur 100, sachant que l'algorithme de Dijkstra ne donnait qu'un seul chemin. Enfin pour 20 cas, IOGA a trouvé plusieurs chemins (entre 3 et 8 chemins). L'utilisation des paramètres (SIZE = 10 et TOP = 10) permet d'avoir presque le même temps d'exécution entre les deux algorithmes IOGA et Dijkstra, par contre si nous utilisions TOP = 35, nous aurons un temps d'exécution très élevés.

SIZE	15	15	11	10	10
TOP	35	15	15	10	8
(0) Solution	0	0	0	1	4
(1) Solution	64	64	64	64	63
(2) Solutions	12	12	12	15	18
(3) Solutions	12	12	12	10	7
(4) Solutions	4	4	4	3	1
(5) Solutions	4	4	4	4	4
(6) Solutions	2	2	2	1	1
(7) Solutions	1	1	1	1	1
(8) Solutions	0	0	0	1	1
(9) Solutions	1	1	1	0	0
AVG	1.88	1.88	1.88	1.77	1.65

TABLE 4.4 – Résultats des nombres de solutions

6.2 Graphe non orienté

Pour le graphe non orienté, nous avons appliqué les quatre algorithmes sur la base de données non orientée à savoir : IOGA, Dijkstra, GA et A*. Le tableau suivant représente les résultats obtenus pour les 5 premiers couples, et la moyenne des 500 couples source/destination sachant que nous avons utilisé les paramètres suivantes (TOP = 6 et SIZE = 8).

N.Couple	IOGA	Dijkstra	GA	A*
Couple 1	144	146	148	142
Couple 2	215	219	233	218
Couple 3	190	188	193	190
Couple 4	313	308	316	315
Couple 5	204	207	204	198
AVG	209.64	211.6	217.94	213.38

TABLE 4.5 – Comparaison des quatre algorithmes sur les 5 premiers couples

Le tableau (4.5) explique brièvement la comparaison entre les cinq premiers couples. Chaque case représente le plus court chemin récupéré par l'algorithme entre une source et une destination donnée puisque il existe plusieurs couples de source/destination. Nous avons enregistré de bons résultats pour l'algorithme IOGA, par exemple pour le couple 2, IOGA a eu le meilleur résultat avec une valeur de 215, et aussi en moyenne, avec l'algorithme IOGA, nous avons obtenu la meilleure moyenne de 209,64. Avec l'algorithme de Dijkstra nous avons obtenu une valeur moyenne de 211,6. Par l'algorithme A* nous avons eu une valeur moyenne de 213,38 et enfin avec l'algorithme génétique nous avons obtenu une valeur moyenne de 217,94.

Le graphique suivant représente un histogramme comparatif entre les résultats des quatre algorithmes pour les 10 premières couples et par l'utilisation des paramètres suivantes (TOP = 6 et SIZE = 8).

D'après les résultats de l'histogramme, nous remarquons que l'algorithme IOGA a eu le meilleur résultat dans la majorité des cas par rapport aux autres algorithmes, en deuxième position l'algorithme de Dijkstra a eu aussi de bon résultats, mais IOGA donne plusieurs solutions et Dijkstra donne un résultat unique, ceci est un point très important qui caractérise notre algorithme proposé.

6.3 Complexité

Depuis la fin des années 90, la « complexité » est devenue un calcul nécessaire pour prouver la qualité et la rapidité d'un algorithme [S. Maguire, 2006]. Le but de la complexité est de permettre une comparaison directe entre deux ou plusieurs algorithmes selon le temps d'exécution ou plus précisément selon le nombre d'instructions de chaque algorithme. Dans ce qui suit, nous allons calculer la complexité de chacun des algorithmes utilisés dans notre recherche. Le tableau (4.6) représente la complexité de chaque algorithme.

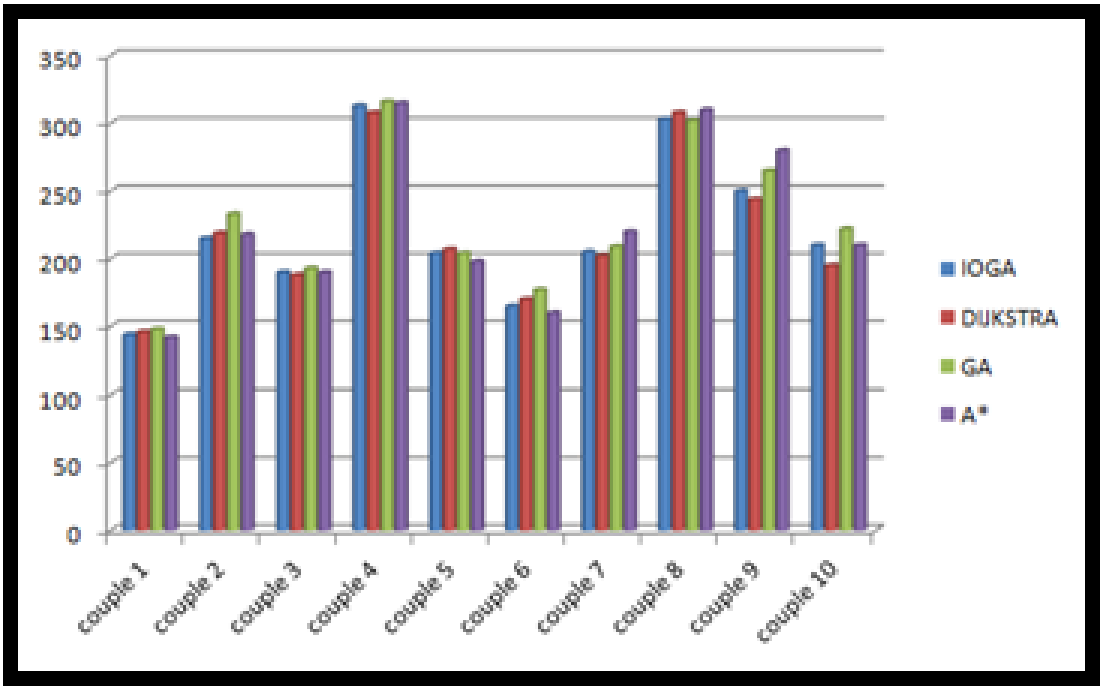


FIGURE 4.7 – Histogramme comparatif des résultats des 4 algorithmes pour les 10 premiers couples

Algorithme	IOGA	Dijkstra	GA	A*
Complexité	O(n * Log(n))	O((a+n) Log n)	O(n * Log(n))	$O(n^2)$

TABLE 4.6 – Complexité des quatre algorithmes

Nous remarquons que l'algorithme de Dijkstra donne la meilleure complexité de O((a + n) log n), l'algorithme IOGA et l'algorithme Génétique en deuxième position avec la même complexité O(n * Log (n)). En dernière position, l'algorithme A* par une forte complexité de O(n2). Nous pouvons donc conclure que l'algorithme de Dijkstra a la meilleure complexité et qu'il reste le meilleur algorithme connu, mais l'algorithme IOGA à l'avantage de récupérer plusieurs chemins avec une complexité un peu plus élevée que l'algorithme de Dijkstra.

7 Conclusion

Les stratégies de la recherche heuristique ont le potentiel d'accélérer considérablement un algorithme de chemin le plus court. Le problème des K-plus courts chemins est à la base de nombreux problèmes d'optimisation combinatoire. Dans notre travail, nous nous sommes concentrés sur le calcul du problème des K-plus courts chemins dans un réseau routier. Le problème des K-plus courts chemins dans lequel on cherche un chemin correspondant au coût minimum d'un nœud d'origine à un nœud de destination dans un réseau sous certaines contraintes.

Une approche d'optimisation améliorée basée sur l'intégration de l'algorithme exact (Dijkstra) et de l'algorithme méta-heuristique (GA) est proposée pour améliorer les performances de calcul des plus courts chemins pour le réseau routier. L'AG est l'une des méthodes méta-heuristiques les plus puissantes. Grâce à l'utilisation d'un algorithme hybride, le temps d'exécution d'IOGA est diminué lors de la résolution du problème des plus courts chemins.

L'IOGA trouve un ou k chemins contenant le coût minimal entre deux sommets dans un grand graphe pondéré orienté et non orienté. Cet algorithme utilise des stratégies qui se sont avérées efficaces pour résoudre le problème des plus courts chemins. Les tests sont effectués à l'aide de réseaux générés aléatoirement. Nous prévoyons que l'algorithme proposé sera très utile pour les études intensives actuelles d'applications réelles de réseaux complexes.

Chapitre 5

Approche contextuelle hybride pour la recommandation des services touristiques

Sommaire

1 Introduction

Aujourd'hui, le tourisme a une grande importance pour l'économie mondiale, il implique la propagation de grandes quantités d'informations [Huang, 2009]. Dans le domaine du tourisme, la recommandation peut être faite dans diverses situations telles que la recommandation d'itinéraire personnalisé [Wang, 2021], la recommandation d'itinéraire intelligent [X. Zhou, 2020], la recommandation de points d'intérêt (POI) [P. Sanchez, 2021] [M. Debnath, 2016], etc.

Au cours de la dernière décennie, plusieurs approches pour les SR ont été proposées pour faire face à ces défis. Malheureusement, les SR actuels (par exemple, les systèmes de recommandation des services touristiques) sont rigides car ils sont complètement isolés des divers concepts de « préférences » et de « contexte » des utilisateurs. Par exemple, une telle rigidité se traduit par des services inadaptés (par exemple, un visiteur peut faire du tourisme de montagne avec un très mauvais climat au lieu de visiter le musée dans ces mauvaises conditions météorologiques). Certains aspects importants doivent être pris en compte pour améliorer le processus de recommandation, comme le contexte d'une destination touristique visitée, les prévisions météorologiques, etc.

L'apprentissage automatique (ML) est un domaine qui tente de faire apprendre aux ordinateurs des modèles sans programmation explicite [S. Kulkarni, 2020]. L'avancement rapide du ML a permis un nouveau paradigme de prise en compte du contexte dans les SR, ce paradigme est donc un ML pour les systèmes de recommandation sensibles au contexte. Cette recherche propose une nouvelle méthode de recommandation basée sur des algorithmes de ML pour améliorer la précision prédictive des SR dans le domaine du tourisme. En effet, un système de recommandation touristique personnalisé devrait comporter les caractéristiques suivantes :

• Il doit extraire les préférences et les intérêts de l'utilisateur en utilisant à la fois des données historiques et actuelles en temps réel.

• Il doit exploiter une partie des caractéristiques de l'utilisateur et des contraintes contextuelles de l'objet de recommandation pour anticiper les résultats et améliorer le processus de matching.

• Il doit également apporter des recommandations contextuelles, déjà adaptées à la situation actuelle de l'utilisateur.

Avec le développement de l'apprentissage automatique, et pour améliorer la précision des systèmes de recommandation, l'objectif principal de l'approche proposée consiste à utiliser des techniques et des algorithmes qui peuvent prédire et proposer des services touristiques pertinents (k-items) aux utilisateurs en fonction de leurs intérêts, besoins ou goûts. Dans cette recherche, nous décrivons comment les techniques d'apprentissage automatique peuvent fournir automatiquement des recommandations personnalisées aux demandeurs en tenant compte à la fois de leurs préférences et de leurs informations contextuelles implicites/explicites et des contraintes contextuelles en temps réel des points d'intérêts.

Motivé par les problèmes mentionnés ci-dessus, contrairement aux approches existantes qui se concentrent uniquement sur les informations spécifiées par l'utilisateur, nous proposons une approche efficace, qui donne aux utilisateurs un moyen adéquat pour prendre en compte leurs préférences explicites/implicites et améliorer la recommandation de service en tirant parti de leurs contextes extraits en temps réel. Dans notre étude de cas, un SR permet d'adapter et de personnaliser la visite de l'utilisateur en fonction d'un contexte, en tenant compte de ses préférences et de ses contraintes. Le touriste sélectionne ce qu'il considère comme ses points d'intérêt (POI), mais certains aspects contextuels doivent être pris en compte pour faire une recommandation contextuelle. En conséquence, pour faire une recommandation contextuelle intelligente, cette recherche propose une approche de recommandation hybride composée à la fois d'un filtrage collaboratif basé sur la mémoire et basé sur un modèle. Le système de recommandation est testé sur quatre algorithmes d'apprentissage automatique les plus utilisés, en particulier Neural Network (NN), K-Nearest Neighbor (KNN), Naïve Bayes (NB) et Random Forest (RF).

Nous construisons un algorithme H-RN efficace et intelligent qui hybride à la fois les algorithmes d'apprentissage automatique les plus connus, à savoir Random Forest et Naïve Bayes. Cette hybridation permet d'extraire et d'exploiter en temps réel les informations relatives au contexte de chaque prestation touristique

candidate et de tirer parti des contraintes contextuelles liées à la localisation de l'utilisateur (afin d'identifier les POI autour de lui) et aux conditions météorologiques de la visite de l'utilisateur, (pour fournir des recommandations plus pertinentes et adaptées à la situation météorologique actuelle). Cependant, jusqu'à présent, aucun des systèmes de recommandation touristique existants n'a étudié l'hybridation ci-dessus. Par conséquent, l'algorithme de recommandation touristique sensible au contexte proposé, basé sur des techniques d'apprentissage automatique, est appliqué aux données extraites de quatre ensembles de données du monde réel, à savoir (i) TripAdvisior, (ii) Dataset tsmc, (iii) Hotel Reviews et (iv) dataset ubicomp afin de comparer l'efficacité des différents algorithmes de recommandation et d'évaluer le système proposé.

Pour évaluer les performances de notre système hybride et les quatre autres systèmes de comparaison, nous avons utilisé plusieurs critères, à savoir (i) Critère d'évaluation avec split validation (Rappel, Précision, F-mesure et Accuracy) sans et avec météo, (ii) Critère d'évaluation avec cross-validation (sans et avec l'aléatoire),(iii) Tests statistiques qui contient deux métriques, One Way ANOVA et la diversité, (iiii) Error metrics qui consite à deux métriques : RMSE et MAE.

L'approche proposée a été développée en utilisant le langage Python et les résultats obtenus montrent que notre proposition hybride donne une amélioration de recommandation par rapport aux autres quatre algorithmes et confirment l'efficacité du nouvel algorithme par contribution au degré de plusieurs métriques de test.

2 Nouvelle approche de recommandation hybride

Le but ultime de notre travail est de développer un système de recommandation de tourisme hybride sensible au profil, aux préférences et au contexte qui recommande des lieux touristiques personnalisés en fonction des préférences de l'utilisateur ainsi que les informations contextuelles liées aux utilisateurs (localisation et informations météorologiques), comme illustré à la figure (5.1). Le système proposé tire parti à la fois des méthodes du filtrage collaboratif basées

sur la mémoire et basées sur des modèles.

L'objectif principal est de construire un modèle de recommandation qui utilise les informations de l'utilisateur telles que le profil, les préférences, l'emplacement (l'emplacement est utilisé pour identifier le POI autour de l'utilisateur) et les informations du service touristique telles que les conditions météorologiques (pour fournir des recommandations plus pertinentes adaptées à la situation météorologique). Cette approche permet de générer les top-N recommandations en utilisant une technique de filtrage hybride et des algorithmes d'apprentissage automatique.

La figure (5.1) représente une architecture globale des informations d'entrées de notre système de recommandation.

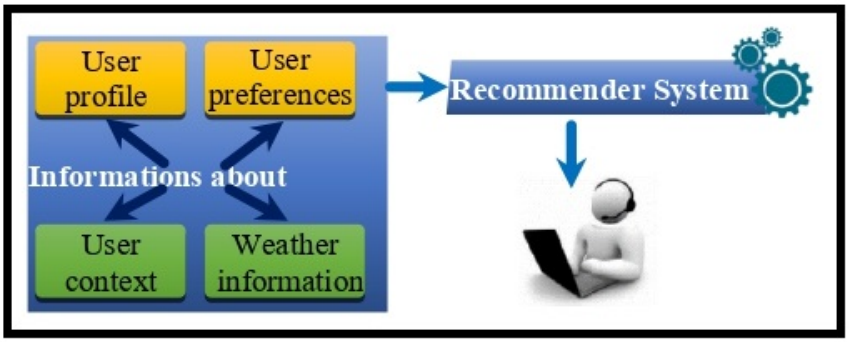


FIGURE 5.1 – Architecture globale de l'approche proposée

L'architecture détaillée de l'approche proposée est illustrée dans la figure (5.2), elle comporte quatre phases principales, à savoir :

(i) la collecte et le pré-traitement des ensembles de données,

(ii) l'extraction des caractéristiques contextuelles,

(iii) l'apprentissage et la prédiction,

(iv) la recommandation en deux étapes : avec et sans situation météorologique.

Comme nous avons expliqué précédemment, l'architecture de notre approche contient quatre phases où chaque phase représente une partie importante du processus de recommandation.

Le pré-traitement des différentes bases de données est la première phase, elle est nécessaire pour avoir des informations fiables et claires et aussi pour avoir une égalité entre les items puisque une information manquante engendre une

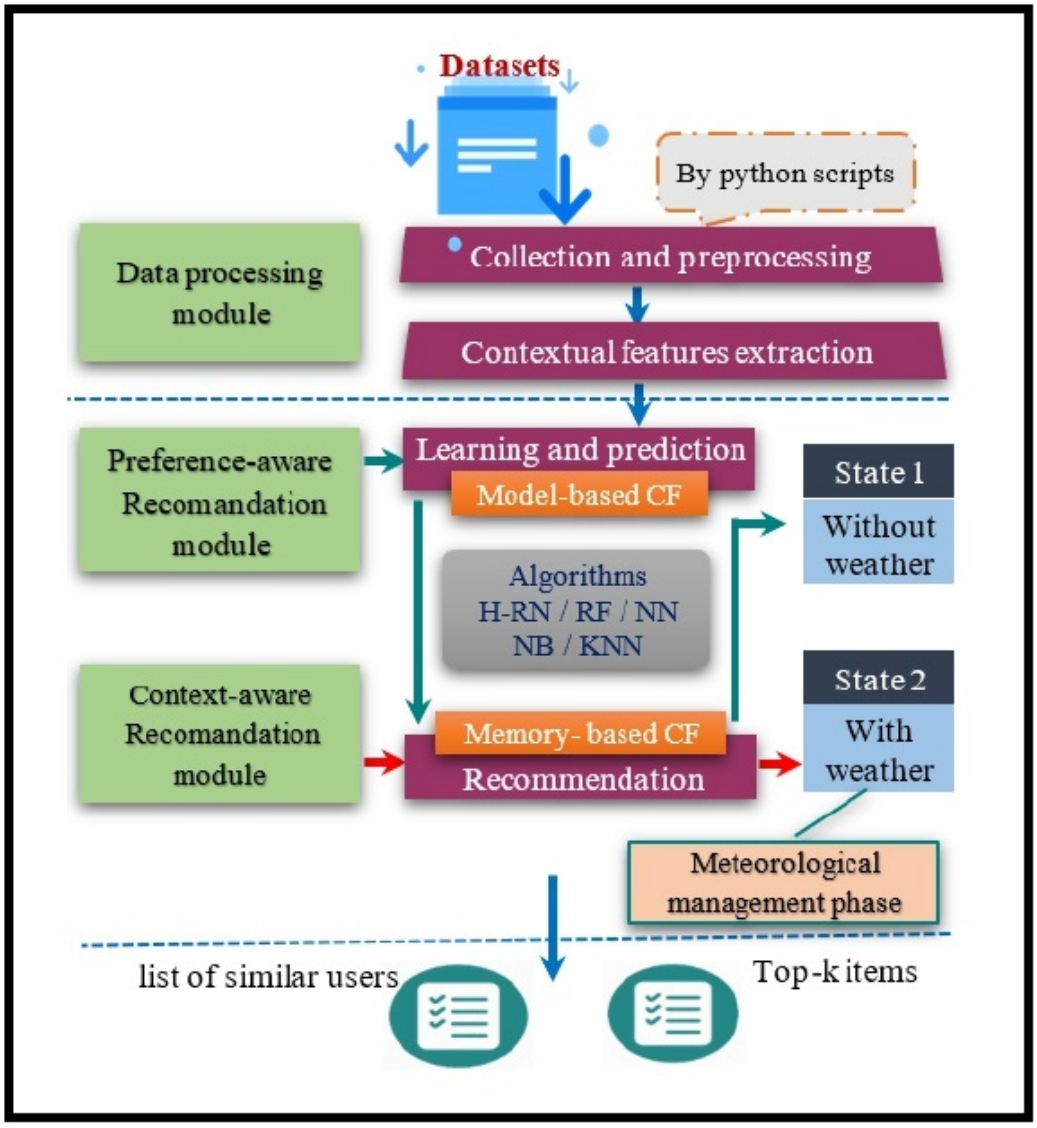


FIGURE 5.2 – Architecture détaillée de l'approche proposée

différence dans les résultats. Cette phase a été implémenté à l'aide de quelques scripts python que nous avons développé. La partie *3 Data-sets et pré-traitement* explique en détail cette phase.

Après avoir préparer les quatre matrices des jeux de données, nous devrons extraire les informations et les caractéristiques contextuelles des bases de données afin d'enrichir le système par des informations permettant de connaître les utilisateurs et les items et les relations entre eux. Soient le profil de l'utilisateur, ses préférences et son contexte, ces informations permettent au système de connaître l'utilisateur d'une façon intelligente et logique, et choisir les meilleurs items qui vont mieux satisfaire le besoin de l'utilisateurs. La partie *3.3 Extraction des caractéristiques contextuelles* explique cette phase.

La troisième phase représente l'apprentissage et la prédiction, l'apprentissage concerne la partie basée sur le modèle où nous appliquons l'algorithme probabiliste (NB) sur nos jeux de données, ensuite nous appliquons l'algorithme de l'arbre aléatoire (RF). Cette dernière partie représente la partie basée sur la mémoire, qui est la dernière phase, le résultat est une liste des items ordonnée prête à recommander aux utilisateurs. Nous expliquons en détail le processus de recommandation et d'hybridation dans la partie *4 Nouvel algorithme hybride H-RN*.

3 Description et pré-traitement des Data-sets

La création d'un nouveau algorithme nécessite toujours une partie de vérification et de validation sur différents échantillons de données afin de tester son efficacité par rapport aux algorithmes existants dans la littérature. Dans notre projet, nous avons choisis quatre bases de données pour évaluer notre algorithme et faire plusieurs tests et comparaisons. Les quatre bases de données sont caractérisées par des critères qui sont différents en termes de taille, de types et de catégories, afin d'avoir une diversité et faire des tests sur des échantillons différents. Nous présentons dans les prochaines lignes les quatre jeux de données et leurs propriétés et caractéristiques, ainsi que leurs sources.

3.1 Description des data-sets

3.1.1 TripAdvisor-dataset-2015

TripAdvisor est un site Web très utilisé par les touristes, il contient des millions de destinations touristiques. D'après [1], les clients de TripAdvisor en 2019 ont fait plus de 66 millions avis avec 53% de destinations en Europe et 23% en Amérique du nord. Nous remarquons que la quantité d'informations disponibles sur TripAdvisor est importante et très riches, ce qui a permet de faire une bonne analyse et de tests pertinents des systèmes de recommandation [F. Celli, 2014] [A. Roshchina, 2015].

Nous avons utilisé la version de TripAdvisor 2015 puisque elle contient plusieurs informations qui aident le système de recommandation à bien recommander les destinations touristiques. Ce data-set comporte quatre fichiers suivants :

*/ Le premier fichier "review_32618.xlsx", il comprend une description détaillée des profils des utilisateurs, sachant que chaque utilisateur possède les informations suivantes :

- nom d'utilisateur.
- tranche d'âge ("13-17", "18-24", "25-34", "35-49", "50-64", "65+").
- genre (féminin ou masculin).
- emplacement
- ville, pays.
- style de voyage (hôtel, restaurant, attraction, etc.).
- l'un des 6 types de badges de reviewer disponibles.
- date d'inscription sur TripAdvisor.

*/ Le deuxième fichier "PersScores1098.xlsx", il contient les scores de personnalité pour chaque profil d'utilisateur, ce fichier contient les informations suivantes :

- nombre total d'avis d'hôtel rédigés.
- nombre total d'avis de restaurants écrits.
- nombre total d'avis écrits sur les attractions.

1. https ://www.my-hotel-reputation.com/chiffres-cles-tripadvisor-2019/

- nombre total d'avis spécifiques à une langue rédigés par l'une des premières personnes à avoir évalué un lieu particulier (badge Explorer).

*/ Le troisième fichier "varticles159.xlsx", il contient des échantillons de 5 critiques de texte ou plus (pour chaque utilisateur). Ce fichier contient les quatre informations suivantes :

- nombre total de notes attribuées
- nombre total de photos
- nombre total de messages sur le forum
- nombre total d'articles publiés

*/ Le dernier fichier "user_full.xlsx", il possède le contenu textuel d'un article (disponible uniquement pour certains utilisateurs) et il contient les informations suivantes :

- nombre de villes visitées (badge de passeport)
- total des points TripAdvisor gagnés
- niveau du contributeur TripAdvisor
- nombre total d'avis utiles

La base de données est open source et elle est téléchargeable [1].

3.1.2 Dataset-tsmc2014

Cet ensemble de données contient des données d'enregistrement à long terme (environ 10 mois) à New York et à Tokyo recueillies auprès de Foursquare du 12 avril 2012 au 16 février 2013.

Il contient deux fichiers au format csv [2]. Chaque fichier contient 9 colonnes, qui sont :

1. ID utilisateur
2. ID de lieu
3. ID de catégorie de lieu
4. Nom de la catégorie de lieu

1. http ://twin-persona.org/resources
2. https ://github.com/yepyao/icdm2015/blob/master/data/dataset-tsmc2014/dataset-TSMC2 014-NYC.txt

5. La note

6. Latitude

7. Longitude

8. Décalage du fuseau horaire en minutes (le décalage en minutes entre le moment où cet enregistrement s'est produit et la même heure en UTC)

9. Heure UTC

Le fichier dataset-TSMC2014-NYC.txt contient 227 428 enregistrements à New York, et le fichier dataset-TSMC2014-TKY.txt contient 573 703 check-ins à Tokyo [A. Yang, 2015].

3.1.3 Hotel-Reviews

C'est une base de données qui a été créée en 2017, elle contient plus de 500 000 notes et commentaires d'hôtels. Elle est open source et disponible sur Internet [1]

Nous avons les informations suivantes sur chaque hôtel :

- Adresse de l'hôtel
- Nom de l'Hôtel
- Tags (mots clés sur l'hôtel)
- Jours depuis l'examen
- Latitude et longitude

3.1.4 dataset-ubicomp2013

Cette base de données contient des enregistrements des restaurants de New York city collectés depuis Foursquare du 24 octobre 2011 au 20 février 2012. Elle contient trois fichiers au format csv, dont 3112 utilisateurs et 3298 sites avec 27149 enregistrements et 10377 conseils. Ce jeu de données contient les informations suivantes :

- ID d'utilisateur
- ID du lieu
- Texte de l'info (commentaire).

1. https ://www.kaggle.com/harshmehta6711/hotel-reviews.

- La note

- L'ensemble de balises des lieux correspondants.

Ce data-set est en open source, il est disponible sur internet [1].

3.2 Pré-traitement des data-sets

Cette étape présente le pré-traitement des données, qui consiste à préparer nos données avant de les importer dans le modèle créé, puisque nous devrons toujours préparer nos données pour inclure les informations relatives au profil, aux préférences et au contexte dans le processus de recommandation. L'objectif de cette phase est de :(i) éviter des erreurs de programmation car souvent nous rencontrons le problème de l'absence d'une variable ou d'une case vide dans la BDD ce qui engendre des erreurs, (ii) avoir des data-sets clairs, et (iii) avoir une égalité entre les items dans le domaine de la recommandation, car, par exemple si un utilisateur n'a pas saisie une information quelconque, le système de recommandation peut considérer que la variable égal à zéro, dans ce cas, nous aurons des résultats non fiables et pas d'égalité entre les utilisateurs.

À cette fin, des expériences sont menées sur quatre ensembles de données largement utilisés et accessibles au public, à savoir TripAdvisor dataset-2015, Dataset-tsmc, Hotel-Reviews et dataset-ubicomp que nous avons bien expliqué précédemment.

Pour réaliser le pré-traitement et le nettoyage de toutes ces bases de données, des scripts python sont créé. Le nettoyage et le pré-traitement du jeu de données s'effectuent en quatre étapes, qui se résument comme suit :

1. Suppression des doubles colonnes ou des colonnes inutiles (par exemple comme la date d'ouverture ou le prix d'entrée) ou des lignes qui contiennent au moins une case ou une information manquante par un code python que nous avons créé. Ce code permet de vérifier ligne par ligne toutes les bases de données utilisée dans l'approche, une fois il trouve une information ou une case vide ou non compréhensible (erreur de lecture), il la supprime ensuite il passe à la prochaine

1. https ://github.com/jtib/log6308/find/master

ligne.

2. Calcul de la note moyenne où le même utilisateur a évalué le même lieu plus de deux fois avec des notes différentes, par exemple :

User U1 a noté le lieu L1 par la note 4/5.

User U1 a noté le lieu L1 par la note 3/5.

Dans ce cas, nous gardons uniquement une seule ligne utilisant la formule suivante :

$$Rating\_Average\_U(P) = \frac{\sum_1^P RatingU(i)}{|P|} \tag{5.1}$$

Donc : *Rating Average U(P)* = (4 + 3) / 2 = 3.5.

Sachant que :

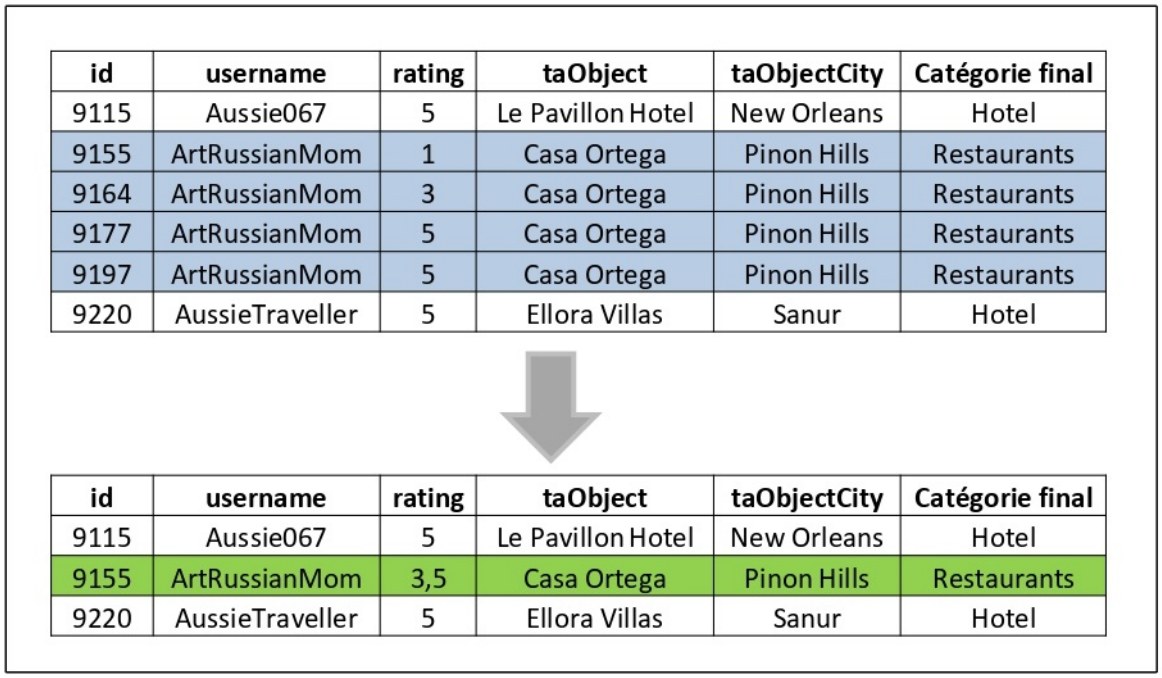
- P : la note estimée de l'utilisateur U pour un ensemble P (plusieurs notes du même utilisateur sur la même place),
- U(i) : représente chaque note donnée par l'utilisateur U pour la place i, et ce dernier appartient à l'ensemble P.

La figure (5.3) représente un exemple de calcul d'une note moyenne dans le cas où un utilisateur a noté le même endroit plusieurs fois.

3. Mise à jour des différents fichiers de la base de données, par exemple si nous avons supprimé un utilisateur dans la première phase, il faut le supprimer dans les autres fichiers de la même base de données, pour qu'on n'aura pas une erreur de programmation par la suite.

4. Ajout de nouvelles informations : dans quelques bases de données, quelques catégories sont très vastes, alors, en utilisant la colonne qui contient des commentaires sur un emplacement quelconque, nous avons fait une catégorisation dans laquelle chaque observation a une catégorie et nous avons ajouté une dernière colonne de catégorie qui contient les attributs suivants : tourisme balnéaire, shopping, monuments, tourisme culturel, lieux de divertissement, tourisme du désert, tourisme de montagne, comme indiqué dans le tableau (5.2).

Les figures (5.4) et (5.5) illustrent un aperçu des utilisateurs de la base de



id	username	rating	taObject	taObjectCity	Catégorie final
9115	Aussie067	5	Le Pavillon Hotel	New Orleans	Hotel
9155	ArtRussianMom	1	Casa Ortega	Pinon Hills	Restaurants
9164	ArtRussianMom	3	Casa Ortega	Pinon Hills	Restaurants
9177	ArtRussianMom	5	Casa Ortega	Pinon Hills	Restaurants
9197	ArtRussianMom	5	Casa Ortega	Pinon Hills	Restaurants
9220	AussieTraveller	5	Ellora Villas	Sanur	Hotel

id	username	rating	taObject	taObjectCity	Catégorie final
9115	Aussie067	5	Le Pavillon Hotel	New Orleans	Hotel
9155	ArtRussianMom	3,5	Casa Ortega	Pinon Hills	Restaurants
9220	AussieTraveller	5	Ellora Villas	Sanur	Hotel

FIGURE 5.3 – Exemple du calcul de la note moyenne d'un même utilisateur sur le même item

données TripAdvisor-dataset-2015 avant et après la partie de pré-traitement. Dans ce jeux de données, nous avons changé les noms des utilisateurs par un nom d'utilisateur standard « Touriste » avec un id ordonné car ce sont des informations confidentielles.

Prenons l'exemple de la première base de données *TripAdvisor-dataset-2015*, pour les deux fichiers *Review 32618.xlsx* et *user full.xlsx*, nous avons trouvé 8476 valeurs d'éléments manquantes, et 2,2 % de valeurs nulles. Ce qui montre l'efficacité de la partie de pré-traitement.

3.3 Extraction des caractéristiques contextuelles

Pour fournir des informations importantes nécessaires à l'évaluation des performances, notre système utilise : (i) des informations de profil (informations personnelles des utilisateurs tels que l'âge, le genre, etc.), (ii) les préférences de l'utilisateur qui peuvent être acquises de deux manières, soit par une liste conte-

	id	Rating	taObject	taObjectCity	Category
0	12160	4.0	Pullman Aachen Quellenhof	Aachen	Hotel
1	11911	5.0	Aamby Valley City	Amby Valley City	Hotel
2	974	3.0	Radisson Blu Scandinavia Hotel, Aarbus	Aarhus	Hotel
3	11815	4.0	Best Western Salford Hall Hotel	Abbots Salfords	Hotel
4	10686	4.0	Midori Japanes Restaurant	Abbotsford	Restaurents
...	...	...	...	...	...
14815	48	5.0	Thurnhers Alpenhof	Zurs	Hotel
14816	10789	4.0	Grizieks Sirtaki	Zwolle	Restaurents
14817	10790	4.0	Bilderberg Grand Hotel Wientjes	Zwolle	Hotel
14818	982	5.0	Domaine PInsPaul	Eze	Hotel
14819	2983	5.0	Eza Vista	Eze	Hotel

FIGURE 5.4 – Capture de TripAdvisor-dataset-2015 avant la partie de pré-traitement

	Username	Age	Gender	Location	TravelStyle
1	Touriste1	25-34	Female	Bristol	Foodie, Nature Lover, Urban Explorer
7	Touriste2	35-49	Male	Wappingers Falls, NewYork	Foodie, Like a local, Trendsetter, Luxury travel
12	Touriste3	25-34	Male	Huddersfield, United Kingdom	Foodie, Beach Goer, ThriftyTraveller, Nightli...
15	Touriste4	25-34	Male	London, United Kingdom	Vegitarian, Urban Explore, Art and Architecture
17	Touriste5	18-24	Female	Melbourne, Australia	Beach Goer, Vegitarian, Urban Explore
...	....	...	...	...	...
7024	Touriste4118	50-64	Male	Paris, France	Foodie, Like a local, Urban Explore, Luxury travel
7028	Touriste4119	35-49	Male	UK	Foodie, Like a local, Urban Explore, Family H...
7029	Touriste4120	35-49	Female	Baltimore, Maryland	Like a local, Thrifty, Art and Architecture
7031	Touriste4121	35-49	Male	Singapore, Singapore	Foodie, Thrifty, Shopping fanatic
7032	Touriste4122	18-24	Male	Hong Kong, China	Foodie, Trendsetter, Beach Goer, Urban Explore...

FIGURE 5.5 – Capture de TripAdvisor-dataset-2015 après la partie de pré-traitement

nant des préférences ordonnées ou par une table d'articles à laquelle l'utilisateur associe une note, sachant que l'ensemble des catégories les plus populaires sont (restaurants, hôtel, loisirs, shopping, tourisme culturel, etc.), (iii) les informations contextuelles à savoir la localisation et la météo. Les informations de profil et de préférences ont été déjà extraites explicitement dans la partie de pré-traitement des bases de données. En ce qui concerne les informations contextuelles, la localisation est représentée par la position GPS de l'utilisateur (latitude et longitude), la situation météorologique peut être obtenue à partir de n'importe quel point GPS à un moment donné en utilisant la bibliothèque python METEO. Cette bibliothèque donne des résultats exacts et rapides. C'est une bibliothèque qui fonctionne avec des données météorologiques.

4 Nouvel algorithme hybride H-RN

Au cours de la dernière décennie, les algorithmes hybrides sont largement utilisés dans pas mal de domaines, et ils ont donné de bons résultats dans la recherche. L'hybridation est une agrégation entre deux ou plusieurs algorithmes, en prenant les forces d'un algorithme et en éliminant ses faiblesses.

4.1 Pseudo code de H-RN

Comme décrit ci-dessus, l'algorithme hybride (H-RN) que cette étude propose prend les forces et élimine les faiblesses des algorithmes de Random Forest et Naïve Bayes à hybrider. Pour construire cet algorithme, nous procédons comme suit :

Nous avons utilisé une approche basée sur la mémoire puisque dans ce type d'approche, le voisinage d'un utilisateur donné est obtenu en calculant ses similitudes avec d'autres utilisateurs. Dans une telle approche, le voisinage ne peut pas être calculé qu'après avoir connu qui est cet utilisateur, ainsi, dans un FC basé sur la mémoire, nous utilisons des données d'évaluation des utilisateurs pour calculer la similitude entre les utilisateurs ou les éléments. Cette approche se base sur les recommandations des FC basées sur la mémoire et les recommandations

Algorithm 10 H-RN algorithm

```
1. Input
2.   Dataset
3.   Weather-table
4. Output
5.   List of items to recommend
6.   List of similar users
7. Begin
8. // apply the preprocessing phase to all the rows of the dataset
9. Dataset ← Preprocessing(Dataset)
10. // apply the Contextual features extraction phase on the database
11. Dataset ← Features-extraction(Dataset)
12. // apply the H-RN function on the dataset
13. H-RN (Dataset, Weather-table)
14. Function H-RN (Data, Weather-table)
15.     Begin Function
16.     // Read all the informations of New-user
17.     New-user ← read (Age, Sex, GPS-position, Preference, ...)
18.     // Apply Naïve Bayes algorithm
19.     For each line1 in Data
20.         For each case in New-user
21.             Result ← distance between current line and new-user
22.         End for
23.         Distance.append (Result)
24.     End for
25.     NB-Ordered-list ← Ascending-sort (Distance)
26.     //(Model based CF)
27.     Tree ← Generate-TRF (NB-Ordered-list)
28.     // TRF = Tree Random Forest (Memory based CF)
29.     // Apply the Random Forest algorithm on the tree
30.     R-score ← Random-Forest (Tree)
31.     // weather phase
32.     Weather-answer ← False
33.     Write ("if you want to calculate the weather phase, type True, otherwise
type False")
34.     Read ("Weather-answer")
```

```
35.    If (Weather-answer=True) then
36.      For each line2 in Data
37.          W-score ← Weather-phase (Data, Weather-table)
38.        overall-score ← R-score + W-score
39.      End for
40.    End if
41.    // Recommend
42.    For i in range (0, K)
43.      Recommend the highest ranked i overall-score
44.    End for
45.    Return (overall-score)
46.    End Function
47. End
```

des top-N basées sur les items/user.

Nous calculons la similarité entre les utilisateurs pour recommander k items pour le nouveau utilisateur sachant que nous utilisons seulement la similarité user/user, et même si nous utilisons la similarité item/item, nous restons toujours dans FC basés mémoire.

Les avantages de cette approche incluent : l'exploitabilité des résultats, qui est un aspect important des systèmes de recommandation ; création et utilisation faciles ; facilité d'avoir de nouvelles données ; l'indépendance du contenu des éléments recommandés ; bonne mise à l'échelle avec des éléments co-évalués.

4.2 Fonctionnement détaillé de H-RN

Dans notre approche, nous avons hybridé naïve Bayes et Random Forest en un seul algorithme H-RN. Ce dernier est basé sur le calcul des utilisateurs les plus proche en utilisant l'algorithme Naïve Bayes, Pour la partie recommandation finale, H-RN utilise l'algorithme de Random Forest.

La première partie de H-RN consiste à lire le data-set ainsi que les informations du nouveau utilisateur (âge, genre, position GPS, préférence, etc.) à qui nous recommandons les K-items à la fin de l'algorithme.

Nous utilisons l'algorithme naïve bayes sur la partie modèle qui est un algorithme probabiliste très efficace. Et il est connue par la formule suivante :

$$P(y|X) = \frac{p(X|y)P(y)}{P(X)} \tag{5.2}$$

• y représente le nombre d'éléments visités par le nouvel utilisateur ;

• X représente le nombre d'éléments visités par tous les utilisateurs de la partie modèle ;

• P (y/X) représente la probabilité du nouvel utilisateur par rapport à tous les utilisateurs de la partie modèle ;

• P (X/y) représente la probabilité de tous les utilisateurs de la partie modèle par rapport au nouvel utilisateur ;

• P (y) représente la probabilité du nouvel utilisateur ;

• P (X) représente la probabilité des utilisateurs de la partie modèle.

Nous calculons la distance entre le nouvel utilisateur et les utilisateurs existants dans l'ensemble d'apprentissage du jeu de données en utilisant la formule suivante :

$$Distance(Ai,Uj) = \sqrt{(V1Ai - V1Uj)^2 + (V2Ai - V2Uj)^2 + ... + (VnAi - VnUj)^2} \tag{5.3}$$

Où Ai est le nouvel utilisateur, Uj est l'ensemble d'entraînement des utilisateurs et V représente les informations des utilisateurs. Toutes ces variables sont représentées dans le tableau suivant :

Variables	V1	V2	V3	.	.	VN
nouvel utilisateur	V1A	V2A	V3A	.	.	VNA
utilisateur 1	V1U1	V2U1	V3U1	.	.	VNU1
utilisateur 2	V1U2	V2U2	V3U2	.	.	VNU2
.	.	.	.	.	.	.
.	.	.	.	.	.	.
utilisateur M	V1UM	V2UM	V3UM	.	.	VNUM

TABLE 5.1 – Tableau explicatif de la relation user/item

les résultats obtenue sont mette dans un tableau qui s'appelle **Distance** et qui

contient N lignes où N représente le nombres des utilisateurs de la partie modèle. Et il contient deux colonnes, la première contient le nom de l'utilisateur (ou l'Id) et la deuxième contient le score qui représente la distance entre l'utilisateur de cette ligne et le nouveau utilisateur.

Nous ordonnons le tableau **Distance** par la fonction **Ascending sort** dans le tableau **NB Ordered list**, dès qu'on aura un tableau des distances bien organisé, nous utilisons la fonction **Generate TRF** qui convertit le tableau **NB Ordered list** en un arbre qu'on met dans la variable **Tree**, sachant que si nous convertissons le tableau **Distance** qui n'est pas ordonnée nous n'aurons pas les même résultats.
Finalement et après avoir l'arbre, nous appliquons directement l'algorithme forêt aléatoire sur l'arbre, ce dernier va générer plusieurs arbres similaires aléatoirement et par plusieurs itération. Dans notre cas, nous avons fait 20 itérations et nous avons généré 30 arbres dans chaque itération, pour que H-RN sélectionne à la fin les meilleures K items à proposer et à stocker dans le tableau **RF Ordered list**, où K=5.

Afin d'analyser les performances du système, de nombreuses expérimentations sont menées à l'aide de quatre algorithmes de machine learning (Random Forest, Naïve Bayes, Réseau neuronal et KNN) et notre algorithme hybride H-RN, appliqués à deux étapes de recommandation différentes, à savoir :

• Étape de recommandation sensible aux profil, aux préférences et à l'emplacement.

• Étape de recommandation des POI en fonction de la météo.

4.3 Phase de gestion météorologique

La sensibilité au contexte est un point important dans notre approche. Le système de recommandation va recommander des items d'après le contexte de l'utilisateur. Le contexte est représenté par la position GPS, ou la nature de l'endroit où il se trouve, par exemple si quelqu'un se trouve à la wilaya de Tlemcen ou même il est proche de cette ville, nous lui recommandons de visiter les meilleurs endroits de l'art andalou.

Nous avons amélioré les performances de notre système de recommandation en ajoutant la situation météorologique qui représente un deuxième point important de la sensibilité au contexte. En général, avant de programmer une sortie, il faut penser avant tout à la situation météorologique, si elle est adaptée à l'endroit qu'on souhaite visiter ou non. Pour parvenir à un tel résultat, notre système récupère la situation météorologique de n'importe quel point GPS à n'importe quel moment donné par l'utilisateur, à partir de la bibliothèque python METEO. Cette dernière est très rapide et donne des résultats exactes. En utilisant les informations retournées de cette phase, nous obtenons un score qui représente un pourcentage de calcul entre la météo et la destination. C'est-à-dire que si le temps est beau et ensoleillé, le score d'une plage sera très élevé, et si le visiteur veut faire du sport de montagne mais la température est très élevée, alors le score, dans ce cas, est très faible puisque l'utilisateur ne peut pas faire du sport dans une telle condition météorologique. Le tableau suivant (5.2) explique comment nous avons attribué le degré de ce score et la relation entre les points d'intérêts et la météo, sachant que les deux catégories *Hôtels* et *Restaurants* sont considérés comme acceptables pour les différentes situations météorologique, car la météo n'a aucun effet sur ce genre de catégories.

Conditions météorologiques	Température			Humidité			Météo				
types des items	LT	BT	HT	LH	BH	HH	SW	CA	R	HR	S
Tourisme balnéaire	22	32	42	30%	60%	90%	95%	60%	05%	00%	00%
Shopping	10	20	30	30%	60%	90%	70%	80%	50%	30%	05%
Monuments	10	20	30	30%	60%	90%	70%	80%	50%	30%	05%
Tourisme culturel	10	20	30	30%	60%	90%	70%	80%	50%	30%	05%
Lieux de divertissement	10	25	40	40%	60%	80%	70%	80%	50%	30%	05%
Tourisme du désert	05	20	35	50%	60%	70%	30%	70%	50%	30%	05%
Tourisme de montagne	05	16	27	40%	60%	80%	80%	60%	40%	20%	70%

TABLE 5.2 – Corrélation entre les caractéristiques météorologiques et les lieux touristiques

sachant que :

LT représente Low Temperature

BT représente Best Temperature
HT représente High Temperature
LH représente Low Humidity
BH représente Best Humidity
HH représente High Humidity
SW représente Sunny Weather
CA représente Clouded Atmospher
R représente Rain
HR représente Heavy Rain
S représente Snow

A partir du tableau précédent, nous avons trois types d'informations à récupérer : la température, l'humidité et les conditions climatiques, chacune représentant 33 % de la note météo finale. Pour calculer le score partiel pour chaque information, nous devons calculer la distance entre les données récupérées en temps réel et les meilleures informations (température, humidité et la situation du météo) selon le tableau (5.2) en utilisant l'équation de distance de Manhattan comme décrit dans les formules suivantes :

$$DisTem = 100 - 100/(BT - LT) * |BT - TR| \tag{5.4}$$

La formule (5.4) représente la distance de la température sachant que **TR** est la température en **temps réel**.

$$DisHum = 100 - 100/(BH - LH) * |BH - HR| \tag{5.5}$$

La formule (5.5) représente la distance de l'humidité sachant que **HR** est l'humidité en **temps réel**.

$$ScoWea(Wri, Wbj) = \sqrt{\frac{(DisTem + DisHum + DisWea)}{3}} \tag{5.6}$$

La formule (5.6) représente la distance entre l'état de la météo en **temps réel** et la meilleure situation météorologique pour une catégorie d'une destination donnée, sachant que **DisWea** représente la distance de la météo que nous récupérons

directement du tableau (5.2).

L'algorithme 11 traite des requêtes météo de la situation météorologique d'une ville ou d'une position GPS, nous avons utilisé un script python qui récupère la situation météorologique (les trois types d'informations que nous avons expliquées précédemment) par JSON depuis un API spécial de météo.

Algorithm 11 situation météorologique en temps réel

```
import requests
import credentials
import re
cities = ["London,uk", "Porto,pt", "Paris,fr"]
T = ["31","08","2021","10","30"]
weatherDict =
defcityForecast(city,T) :
response=requests.get("community-open-weather-
map.p.rapidapi.com/forecast")
q=city,
headers="X-RapidAPI-Host"
"X-RapidAPI-Key" :credentials.rapidapiKey
returnresponse.json()
```

Après avoir calculé les similarités du profil, des préférences et les scores de distance et de la météo, nous calculons le score final. Étant donné que nous avons deux phase, sans météo et avec météo, nous avons deux formules de calcul :

La formule (5.7) représente le score final de la phase sans météo.

$$ScoPh1 = \sqrt{\frac{(SimPro + SimPre + ScoDis)}{3}} \tag{5.7}$$

La formule (5.8) représente le score final de la phase avec météo.

$$ScoPh2 = \sqrt{\frac{(SimPro + SimPre + ScoDis + ScoWea)}{4}} \tag{5.8}$$

Sachant que :

SimPro représente la similarités du profil

SimPre représente la similarités du préférence

ScoDis représente le score de distance calculer par IOGA

ScoWea représente le score de la météo que nous avons expliqué dans la formule (5.6).

5 Résultat et discussion

Pour évaluer notre système, différentes bibliothèques, environnements et langages de développement sont utilisés comme Anaconda, Python, Panda, Scikit-learn et Numpy. Une préoccupation pertinente dans l'approche proposée est la comparaison des algorithmes d'apprentissage automatique largement utilisés dans notre travail. Ce problème peut être résolu en considérant quatre phases :

(i) Critère d'évaluation avec split validation.

(ii) Critère d'évaluation avec cross-validation.

Sachant que les mesures d'évaluation utilisées sont connues et très utilisées dans le domaine de la recommandation [Powers, 2011], notamment le rappel, la précision, F-mesure et l'accuracy.

(iii) Métriques statistiques [Demsar, 2006] qui se composent de deux types de test : tests paramétriques tels que le test T apparié, ANOVA (Analyse de la Variance), etc., et tests non paramétriques tels que le test de Wilcoxon et le test de Friedman. Dans notre cas, nous avons calculé ANOVA et la Diversité.

(iv) Métriques d'erreur [Botchkarev, 2019] qui sont couramment utilisées pour évaluer et tester les performances d'un modèle de régression tel que l'erreur quadratique moyenne (RMSE) et l'erreur absolue moyenne (MAE).

Les quatre ensembles de données décrits dans la section (3.1) sont utilisés pour évaluer les performances de la technique proposée en comparant les résultats obtenus à l'aide des algorithmes RF, NB, NN et KNN avec notre algorithme proposé H-RN. Au total, 26 expérimentations sont menées sur chacun des quatre ensembles de données en utilisant les cinq algorithmes *avec conditions météorologiques/sans conditions météorologiques.*

Nous commençons par l'analyse empirique des résultats avec deux techniques de validation (split validation et cross-validation) et l'étude de certaines métriques d'évaluation populaires pour évaluer les performances de notre système.

Les tests statistiques sont également une partie essentielle qui permet de tirer des conclusions fiables. Le test statistique paramétrique utilisé pour la comparaison de plusieurs algorithmes sur plusieurs jeux de données est le test ANOVA à un facteur, de plus, la diversité, la RMSE et la MAE sont utilisées pour évaluer l'accuracy de la méthode de recommandation suggérée. Ces paramètres, qui sont décrits dans la sous-section suivante, sont largement utilisés pour évaluer la précision de notre approche.

5.1 Critère d'évaluation avec split validation

Split validation est une technique utilisée pour évaluer les performances d'un modèle d'apprentissage automatique, d'une classification ou d'une régression [Y. Xu, 2018]. Un ensemble de données est divisé en deux sous-ensembles : un ensemble de données d'entraînement et un ensemble de données de test. Dans cette approche, chaque ensemble de données a été divisé au hasard en 67 % (ou 21745 lignes pour le premier ensemble de données, 151619 lignes pour le deuxième ensemble de données, 343825 pour le troisième ensemble de données et 56344 pour le dernier ensemble de données) en tant qu'un ensemble d'apprentissage et 33 % (ou 10873 lignes pour le premier jeu de données, 75809 lignes pour le deuxième jeu de données, 171913 pour le jeu de données numéro trois et 28172 pour le dernier jeu de données) comme jeu de test.

Nous utilisons les données de modèle pour apprendre et construire un modèle et les données de test pour évaluer notre système. La séparation des données d'apprentissage et de test est effectuée à l'aide de la méthode "data-split". La figure suivante montre le script python qui permet de diviser les données en modèle et en test.

Avec :

• 67 % de l'ensemble d'apprentissage est représenté en deux parties : *x-train* et *y-train*, les deux dernières parties représentent en général la partie modèle de deux algorithmes x et y dans le cas de la comparaison entre deux algorithmes. Dans notre cas, *x-train* et *y-train* sont les mêmes puisque nous voulons comparer

```
import random
from tkinter import *
[train-data,test-data] = data.split("67%","33%")
#67% training and 33% testing
xtrain=train-data[0]
ytrain=train-data[1]
xtest=test-data[0]
ytest=test-data[1]
```

FIGURE 5.6 – Script python qui permet de diviser les données en modèle et en test

les résultats d'un algorithme avec les résultats réels qui existent dans la base de données et non pas avec un autre algorithme.

• 33 % de l'ensemble de test est également représenté en deux parties *x-test* et *y-test*, *x-test* représente la partie test réel qui existe dans la base de données, et *y-test* représente la partie test récupérée par l'algorithme, cela veut dire que nous espérons que l'algorithme récupère la partie *y-test* identique à *x-test*.

Une meilleure approche avec séparation train/test consiste à utiliser plusieurs ensembles de tests et à calculer la moyenne de la précision réelle, ce qui nous donne une estimation plus précise de la véritable précision d'un modèle sur des données invisibles. L'un des concepts de science des données les plus largement utilisés pour les jeux de tests multiples est connu sous le nom de *"validation croisée"* [Y. Xu, 2018]. Plusieurs manières de découper la rame avec la technique de validation croisée, parmi lesquelles nous citons **k-fold**.

Dans cette section, nous rapportons un ensemble d'expériences menées pour évaluer la méthode proposée de recommandation touristique sensible aux préférences et au contexte. Les figures suivantes montrent les résultats du rappel, de la précision, de la F-mesure et de l'Accuracy avec un tableau du taux moyen de chaque algorithme pour la première phase (sans météo) puis pour la seconde phase (avec météo) par la technique de *split validation*.

5.1.1 Critère d'évaluation avec split validation sans météo

• Rappel

Le rappel est parmi les mesures les plus utilisées dans l'évaluation des systèmes de recommandation, il représente la relation entre le nombre d'items pertinents sélectionnés et le nombre d'items pertinents disponibles dans la base de données. Le tableau suivant représente les résultats du rappel pour la technique split validation sans météo.

Algorithme	Dataset 1	Dataset 2	Dataset 3	Dataset 4
H-RN	0.637	0.657	0.711	0.703
Naïve-Bayes	0.605	0.633	0.653	0.714
Forêt aléatoire	0.703	0.617	0.691	0.688
Réseau neuronal	0.653	0.68	0.707	0.647
KNN (K=5)	0.622	0.583	0.61	0.596
KNN (K=10)	0.57	0.617	0.547	0.572
KNN (K=15)	0.457	0.529	0.511	0.482

TABLE 5.3 – Résultats de Rappel - phase 1 - split validation

D'après le tableau (5.3), nous remarquons clairement que l'algorithme RF donne de bons résultats dans le jeu de données 1. En revanche, H-RN donne le meilleur taux de rappel pour le troisième jeu de données avec une valeur de 0,711. Cette amélioration est due au fait que notre algorithme est basé sur les meilleures parties des deux algorithmes hybrides (RF et NB). De plus, dans le deuxième et le dernier ensemble de données, le taux de rappel de notre algorithme est proche des meilleurs résultats par rapport aux autres algorithmes.

• Précision

La précision représente le nombre d'items pertinents sélectionnés par rapport au nombre total d'items sélectionnés pour une requête donnée, cette mesure permet de détecter la qualité des résultats renvoyés par le système. Le tableau (5.4) représente

les résultats de la précision pour la partie split validation sans météo.

Algorithme	Dataset 1	Dataset 2	Dataset 3	Dataset 4
H-RN	0.647	0.681	0.653	0.711
Naïve-Bayes	0.578	0.705	0.694	0.693
Forêt aléatoire	0.671	0.657	0.603	0.697
Réseau neuronal	0.637	0.67	0.618	0.714
KNN (K=5)	0.641	0.718	0.511	0.571
KNN (K=10)	0.541	0.519	0.491	0.576
KNN (K=15)	0.439	0.476	0.573	0.516

TABLE 5.4 – Résultats de Précision - phase 1 - split validation

La précision de H-RN est proche du meilleur résultat de NB dans les jeux de données 2 et 3. Dans le dernier jeu de données, nous remarquons que la précision de notre algorithme (0,711) est presque similaire au meilleur résultat (0,714) donné par NN. La précision de notre algorithme surpasse la précision de KNN (K = 5, K = 10, K = 15) sur tous les jeux de données sauf le jeu de données 2 où le meilleur résultat de KNN (k = 5) est de 0,718, ce qui n'est pas très loin du meilleur résultat (0,711) de notre algorithme.

• F-mesure

Dans la lignée de la compréhension de la qualité globale d'un système de recommandation, nous combinons le rappel et la précision en utilisant la F-mesure. Le tableau suivant représente les résultats de F-mesure obtenue par les cinq algorithmes pour la partie split validation sans météo.

Algorithme	Dataset 1	Dataset 2	Dataset 3	Dataset 4
H-RN	0.641	0.668	0.68	0.706
Naïve-Bayes	0.591	0.667	0.672	0.703
Forêt aléatoire	0.686	0.636	0.644	0.692
Réseau neuronal	0.644	0.674	0.659	0.678
KNN (K=5)	0.631	0.643	0.566	0.583
KNN (K=10)	0.555	0.563	0.517	0.573
KNN (K=15)	0.447	0.501	0.54	0.498

TABLE 5.5 – Résultats de F-mesure - phase 1 - split validation

Dans ce tableau, nous voyons clairement que H-RN domine l'algorithme KNN (K=5, K=10, K=15) en termes de F-mesure pour tous les jeux de données et les trois autres algorithmes pour les jeux de données 2, 3 et 4 avec une valeur de 0,668, 0,68 et 0,706 respectivement. De plus, H-RN est proche des meilleurs deuxièmes résultats dans le premier ensemble de données, cependant, RF donne la meilleure F-mesure.

• Accuracy

C'est une mesure importante qui permet d'évaluer les performances d'un système, elle représente le rapport entre le nombre total d'items pertinents et le nombre total d'items. Le tableau (5.6) représente l'accuracy entre les quatre jeux de données pour la phase split validation sans météo.

Algorithme	Dataset 1	Dataset 2	Dataset 3	Dataset 4
H-RN	0.678	0.713	0.627	0.671
Naïve-Bayes	0.514	0.637	0.714	0.66
Forêt aléatoire	0.703	0.619	0.653	0.648
Réseau neuronal	0.643	0.704	0.582	0.654
KNN	0.615	0.638	0.543	0.657

TABLE 5.6 – Résultats de l'Accuracy - phase 1 - split validation

Il est facile de remarquer dans tous les jeux de données que H-RN surpasse KNN. De plus, dans les ensembles de données 2 et 4, la prédominance de l'accuracy concerne

le calcul de l'algorithme H-RN, mais dans les deux autres ensembles de données, elle a respectivement un deuxième et un troisième degré d'accuracy.

Nous présentons maintenant le tableau des moyennes qui représente les quatre tableaux précédents, le tableau rassemble la moyenne des résultats de rappel, précision, F-mesure et l'accuracy obtenue par les cinq algorithmes et les quatre jeux de données pour la phase split validation sans météo.

Algorithme	Rappel	Précision	F-mesure	Accuracy
H-RN	0.667	0.673	0.673	0.672
Naïve-Bayes	0.651	0.667	0.658	0.631
Forêt aléatoire	0.674	0.657	0.664	0.655
Réseau neuronal	0.671	0.659	0.663	0.645
KNN (K=5)	0.602	0.61	0.605	0.613
KNN (K=10)	0.576	0.531	0.553	0.613
KNN (K=15)	0.494	0.501	0.497	0.613

TABLE 5.7 – Résultats de la moyenne - phase 1 - split validation

Nous remarquons dans le tableau précédent que l'algorithme H-RN donne les meilleurs résultats en rappel, précision et F-mesure par rapport à KNN (K=5, K=10, K=15). De plus, il donne la meilleure précision avec une valeur de 0,673 et la meilleure F-mesure avec une valeur de 0,673 par rapport à NB, RF et NN. Nous remarquons également que les algorithmes NN et RF donnent un bon taux de rappel par rapport à NB, KNN et H-RN. Aussi nous remarquons que H-RN domine tous les algorithmes pour le critère de l'accuracy.

La figure suivante (5.7) représente quatre histogrammes regroupant et résumant les résultats de la phase 1 (sans météo) de split validation, la figure permet d'analyser les résultats d'une manière globale.

5.1.2 Critère d'évaluation avec split validation avec météo

Comme nous avons expliqué dans le titre précédent les résultats de split validation sans météo, nous expliquons dans les prochaines lignes les résultats de split validation

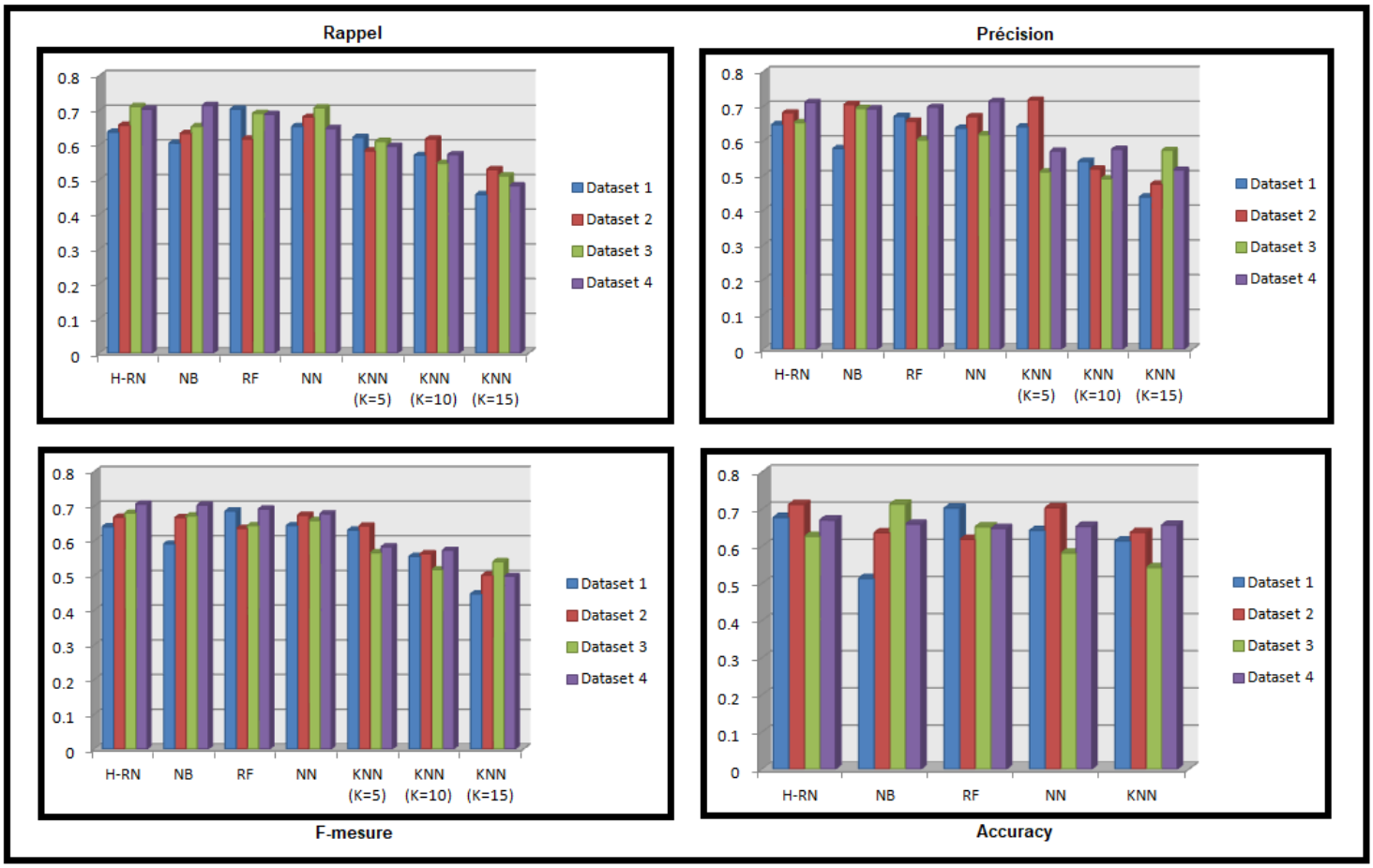


FIGURE 5.7 – Histogrammes des résultats de la phase 1 - split validation

avec météo.

• Rappel

Nous voyons clairement dans le tableau (5.8) que H-RN donne le meilleur rappel par rapport à KNN (K = 5, K = 10, K = 15) dans tous les ensembles de données et le deuxième bon résultat dans les ensembles de données 1, 2 et 4 avec une valeur de 0,742, 0,683 et 0,714, respectivement, et dans le troisième ensemble de données, il est proche des deux premiers meilleurs résultats.

Algorithme	Dataset 1	Dataset 2	Dataset 3	Dataset 4
H-RN	0.742	0.683	0.734	0.714
Naïve-Bayes	0.761	0.654	0.72	0.685
Forêt aléatoire	0.715	0.67	0.741	0.719
Réseau neuronal	0.687	0.712	0.737	0.71
KNN (K=5)	0.672	0.718	0.727	0.685
KNN (K=10)	0.642	0.594	0.609	0.636
KNN (K=15)	0.536	0.548	0.641	0.53

TABLE 5.8 – Résultats du Rappel - phase 2 - split validation

• Précision

Le tableau (5.9) représente les résultats de la précision de split validation avec météo.

Algorithme	Dataset 1	Dataset 2	Dataset 3	Dataset 4
H-RN	0.683	0.711	0.703	0.691
Naïve-Bayes	0.688	0.692	0.712	0.674
Forêt aléatoire	0.71	0.691	0.689	0.707
Réseau neuronal	0.675	0.725	0.704	0.673
KNN (K=5)	0.573	0.651	0.634	0.619
KNN (K=10)	0.579	0.563	0.622	0.583
KNN (K=15)	0.572	0.643	0.581	0.561

TABLE 5.9 – Résultats de Précision - phase 2 - split validation

Dans le tableau (5.9), H-RN fournit la meilleure précision dans les quatre ensembles de données par rapport à l'algorithme KNN (K = 5, K = 10, K = 15). Il donne la deuxième bonne précision dans les jeux de données 2 et 4 avec une valeur de 0,711 et 0,691 respectivement en ce qui concerne NB, RF et NN, mais dans le premier et le troisième jeu de données, il est proche des deux premières meilleures valeurs de précision.

• F-mesure

Le tableau (5.10) représente les résultats de F-mesure pour la phase split validation avec météo.

Algorithme	Dataset 1	Dataset 2	Dataset 3	Dataset 4
H-RN	0.711	0.696	0.718	0.702
Naïve-Bayes	0.722	0.672	0.715	0.679
Forêt aléatoire	0.712	0.68	0.714	0.712
Réseau neuronal	0.68	0.718	0.72	0.691
KNN (K=5)	0.618	0.682	0.677	0.65
KNN (K=10)	0.608	0.578	0.615	0.608
KNN (K=15)	0.553	0.591	0.609	0.545

TABLE 5.10 – Résultats de F-mesure - phase 2 - split validation

Nous remarquons que H-RN donne les meilleurs résultats car il est confiné entre 0,691 et 0,718, par contre, les trois autres algorithmes ont moins de performances, c'est à dire, NB donne des résultats entre 0,672 et 0,722, tandis que NN donne des résultats entre 0,68 et 0,72.

• Accuracy

Le tableau suivant représente les résultats d'Accuracy pour la phase split

validation avec météo.

Algorithme	Dataset 1	Dataset 2	Dataset 3	Dataset 4
H-RN	0.702	0.651	0.563	0.688
Naïve-Bayes	0.675	0.681	0.591	0.632
Forêt aléatoire	0.714	0.61	0.682	0.704
Réseau neuronal	0.694	0.719	0.551	0.653
KNN	0.594	0.622	0.57	0.627

TABLE 5.11 – Résultats de l'Accuracy - phase 2 - split validation

Nous voyons que H-RN domine KNN et NN dans la plupart des ensembles de données, et les résultats sont très proches avec une petite dominance de l'algorithme H-RN sur NB et RF dans les premiers et les derniers ensembles de données. Les algorithmes NN et RF offrent la meilleure accuracy pour les ensembles de données 2 et 3, respectivement.

Le tableau suivant représente la moyenne des résultats de rappel, précision, F-mesure et d'accuracy pour la phase split validation avec météo.

Algorithme	Rappel	Précision	F-mesure	Accuracy
H-RN	0.718	0.697	0.706	0.651
Naïve-Bayes	0.705	0.691	0.697	0.644
Forêt aléatoire	0.711	0.699	0.704	0.677
Réseau neuronal	0.711	0.694	0.702	0.654
KNN (K=5)	0.7	0.619	0.657	0.603
KNN (K=10)	0.62	0.586	0.603	0.603
KNN (K=15)	0.563	0.589	0.576	0.603

TABLE 5.12 – Résultats de la moyenne - phase 2 - split validation

A partir du tableau (5.12), nous observons que l'algorithme H-RN donne les meilleurs résultats en rappel avec 0,718 et F-mesure avec 0,706, de même qu'il est en deuxième position en précision avec 0,797 avec une petite différence de 0,02 par rapport au meilleur résultat de RF. Ce dernier a retourné de bons résultats en Accuracy. De plus, nous remarquons que le H-RN fournit les meilleurs résultats en rappel, précision, F-mesure et en accuracy vis-à-vis du KNN (K=5, K=10, K=15). L'algorithme NN

donne également de bons résultats par rapport à NB et RF.

La figure suivante représente quatre histogrammes regroupant et résumant les résultats de la phase 2 (avec météo) de split validation, cette figure permet d'analyser les résultats de manière globale.

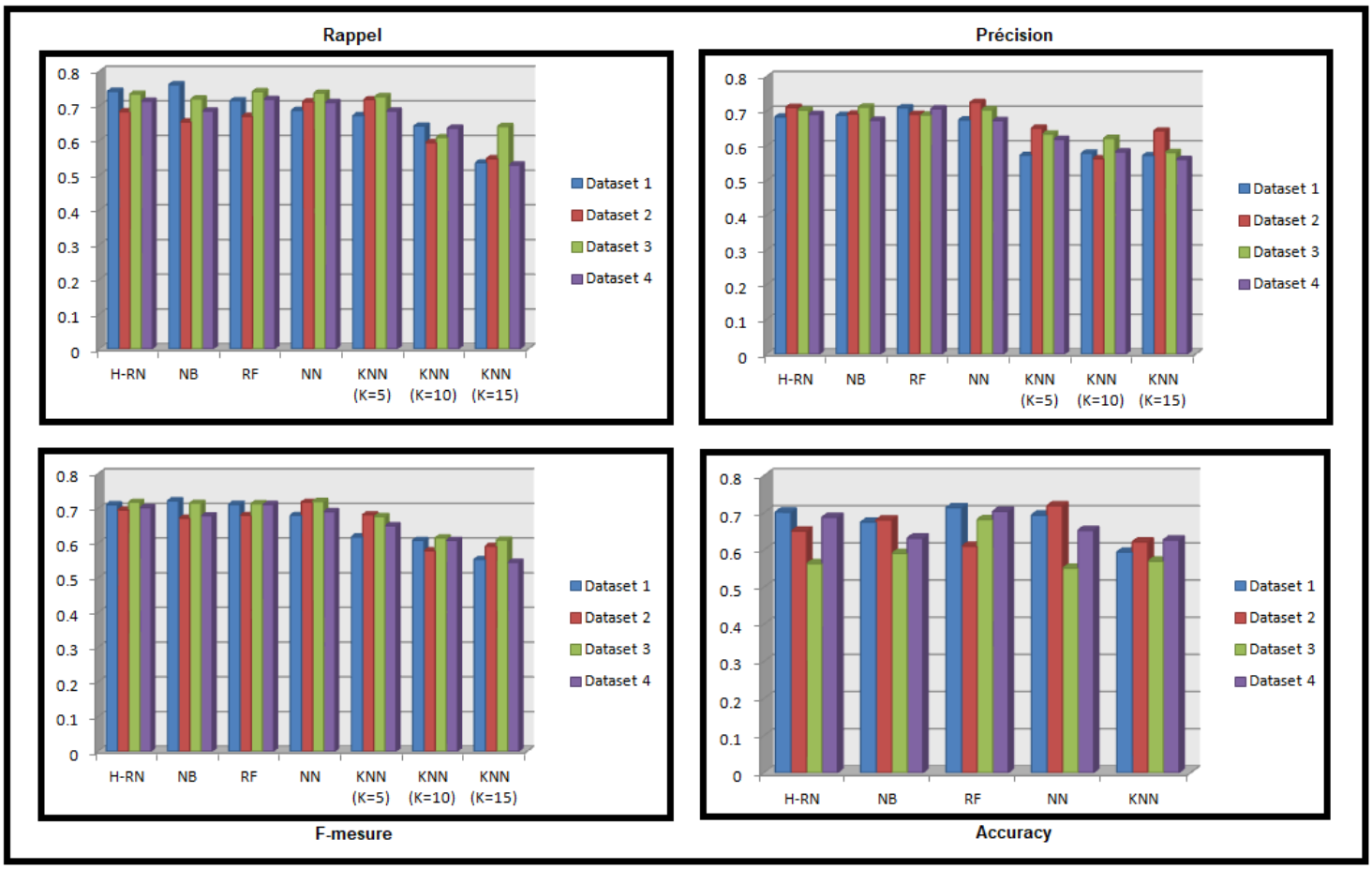


FIGURE 5.8 – Histogrammes des résultats de la phase 2 - split validation

À partir de l'analyse détaillée des résultats expérimentaux du rappel, précision, f-mesure, accuracy et de la moyenne en utilisant les algorithmes NB, RF, NN, KNN et H-RN, nous pouvons conclure qu'il existe deux principaux résultats donnés par notre algorithme hybride :

• Phase 1 (sans météo) : produit des résultats moyen du taux de rappel, de précision et de f-mesure de 67,1 %, et avec une bonne accuracy pour H-RN.

• Phase 2 (avec conditions météorologiques) : produit un taux moyen d'accu-

racy, du rappel, de la précision et de la f-mesure de 70,7 %, qui est augmenté de 3,6 % en raison de l'amélioration du taux de tous les critères d'évaluation avec le degré de rappel le plus élevé (0,718), précision (0,697), F-mesure (0,706) et une amélioration remarquable pour l'accuracy par rapport aux résultats de la phase 1 et également, nous avons une amélioration de la valeur de précision par rapport aux résultats des autres algorithmes dans les deux phases.

Nous pouvons affirmer que les meilleurs scores moyens sont obtenus en intégrant la météo dans la deuxième phase du processus de recommandation et également par la combinaison des algorithmes NB et RF. Techniquement, nous pouvons conclure que les conditions météorologiques peuvent améliorer la précision de la méthode proposée dans les systèmes de recommandation. De plus, H-RN est plus précis que les quatre autres algorithmes (Naïve Bayes, Random Forest, Neural Network et KNN).

5.2 Critère d'évaluation avec cross-validation

Dans la majorité des techniques d'évaluation des systèmes de recommandation, les jeux de données sont divisés en deux parties, un ensemble de données de la partie modèle est utilisé pour former le système de recommandation, et l'ensemble de données de test est utilisé pour prédire les notes et recommande les listes Top-k avec les notes les plus élevées pour le touriste en particulier ou l'utilisateur cible en général.

En cross-validation, l'ensemble de données de modèle est divisé en k sous-ensembles différents (appelés « plis »). Avec la validation croisée k-fold, nous divisons l'ensemble de données complet que nous avons en k parties disjointes de la même taille, c'est-à-dire que nous divisons nos données en "k folds" ou fractionnements d'entraînement/test où k est le nombre de validations croisées à exécuter (par exemple 5). Nous créons ces plis de manière à ce que chaque point de notre jeu de données se produise dans exactement un jeu de test.

La figure (5.9) montre le script python qui permet de diviser les données de la partie modèle et la partie de test avec une validation croisée K-fold (dans ce cas avec k=5). Nous avons utilisé dans nos expériences 67 % des données comme

un ensemble de données d'apprentissage et 33 % comme ensemble de données de test.

```
import pandas as pd
from sklearn.model_selection import KFold
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
data = load_bdd(as_frame = True)
df = data.frame
X = df.iloc[:,:-1]
y = df.iloc[:,-1]
k = 5
X_train=67%
X_test=33%
kf = KFold(n_splits=k, random_state=None)
model = LogisticRegression (solver= 'liblineaR')
acc_score = []
for train_index , test_index in kf.split(X):
   X_train , X_test = X.iloc[train_index,:],
X.iloc[test_index,:]
  y_train , y_test = y[train_index] , y[test_index]
   model.fit(X_train,y_train)
   pred_values = model.predict(X_test)
   acc = accuracy_score(pred_values , y_test)
   acc_score.append(acc)
avg_acc_score = sum(acc_score)/k
print (.format(acc_score))
print(.format(avg_acc_score))
```

FIGURE 5.9 – Script python de la partie validation croisée K-Fold

Nous utilisons une méthode qui impliquait un "paramètre de réglage", un paramètre qui n'est pas estimé, mais juste en quelque sorte deviné. Dans nos expériences, d'abord, nous avons divisé les données en 5 blocs (appelé 5-fold cross-validation) et utilisé liblinear [R. Fan, 2008] qui permet l'estimation de modèles linéaires prédictifs pour la régression. Ensuite, nous avons utilisé une validation croisée 10 fois pour aider à trouver la meilleure valeur pour ce paramètre de réglage. Enfin, nous avons divisé les données en 15 fois.

La validation croisée consiste à entraîner puis à valider notre modèle sur plusieurs coupes possibles de la rame. Par exemple, en divisant la rame en 5 parties, nous pouvons entraîner notre modèle sur les 4 premières parties puis le valider sur la cinquième partie. Ensuite, on va refaire tout sa pour toutes les configurations possibles. Au final, on prendra la moyenne des 5 notes. Et donc, quand on voudra comparer deux modèles alors on sera sûr de prendre celui qui a eu en moyenne les meilleures performances.

Le calcul de l'évaluation de la véritable et la vraie performance du modèle est un élément essentiel de tout algorithme de recommandation. La technique de validation croisée nous permet de comparer différentes méthodes d'apprentissage automatique. Cette section discute des résultats de la deuxième expérience pour mesurer et améliorer la précision des algorithmes prédictifs et montre l'utilité des caractéristiques contextuelles extraites (météo). Cette méthodologie est appliquée sur quatre jeux de données avec cinq algorithmes (H-RN, KNN (N=10, N=15), RF, NN, NB) sans et avec contexte météorologique (phase 1 et phase 2 respectivement) où chaque modèle est formé sur un sous-ensemble des données initiales et des prédictions sont formées pour l'autre sous-ensemble comme détaillé ci-dessous.

Les tableaux suivants résument l'évaluation de la validation croisée pour cinq méthodes d'apprentissage automatique différentes, une fois sans caractère aléatoire où l'évaluation effectuée sans et avec la météo comme dans les tableaux (5.13), (5.14), (5.15) et une fois avec le caractère aléatoire comme dans les tableaux (5.16), (5.17), (5.18). Le caractère aléatoire dans les deux cas est contrôlé via le paramètre random-state, par exemple, le paramètre *random-state* par défaut est *"None"* dans le cas du sans caractère aléatoire, comme décrit dans la figure (5.9).

5.2.1 K-fold cross-validation sans Random

Les trois tableaux suivants représentent l'évaluation des cinq algorithmes de recommandation sur quatre jeux de données, évalués par la technique de validation croisée K-fold sans *Random* avec (K=5, 10 et 15) en deux phases sans et

avec météo.

Algorithme		H-RN	KNN (K=10)	KNN (K=15)	RF	NN	NB
Dataset 1	Phase 1	0.5972	0.5407	0.5216	0.6015	0.5733	0.5582
	Phase 2	0.6048	0.534	0.538	0.6042	0.5981	0.5840
Dataset 2	Phase 1	0.5672	0.5304	0.5281	0.5360	0.5619	0.5426
	Phase 2	0.5992	0.5607	0.5462	0.6017	0.6024	0.5933
Dataset 3	Phase 1	0.5982	0.6076	0.5933	0.5973	0.5810	0.5545
	Phase 2	0.6288	0.6198	0.6088	0.5288	0.6177	0.5952
Dataset 4	Phase 1	0.5841	0.5739	0.5804	0.5267	0.5759	0.5516
	Phase 2	0.5984	0.5967	0.5837	0.5995	0.5980	0.5963

TABLE 5.13 – Résultats de cross-validation sans Random avec K=5

Algorithme		H-RN	KNN (K=10)	KNN (K=15)	RF	NN	NB
Dataset 1	Phase 1	0.5913	0.5207	0.5087	0.6024	0.5683	0.5504
	Phase 2	0.5974	0.5193	0.5248	0.5947	0.5934	0.5895
Dataset 2	Phase 1	0.5584	0.5192	0.5116	0.5631	0.5475	0.5344
	Phase 2	0.5713	0.5273	0.5307	0.5646	0.6071	0.5767
Dataset 3	Phase 1	0.5870	0.5833	0.5544	0.5807	0.5749	0.5562
	Phase 2	0.6072	0.5973	0.5840	0.6166	0.5900	0.5862
Dataset 4	Phase 1	0.5992	0.5720	0.5648	0.5925	0.5408	0.5394
	Phase 2	0.5743	0.5834	0.5609	0.5933	0.5862	0.5907

TABLE 5.14 – Résultats de cross-validation sans Random avec K=10

Algorithme		H-RN	KNN (K=10)	KNN (K=15)	RF	NN	NB
Dataset 1	Phase 1	0.5737	0.5649	0.5420	0.5532	0.5771	0.5436
	Phase 2	0.5737	0.5662	0.5428	0.5628	0.5647	0.5429
Dataset 2	Phase 1	0.5582	0.5533	0.5287	0.5569	0.5560	0.5422
	Phase 2	0.5688	0.5615	0.5473	0.5705	0.5560	0.5585
Dataset 3	Phase 1	0.5609	0.5408	0.5545	0.5573	0.5578	0.5591
	Phase 2	0.5659	0.5482	0.5307	0.5631	0.5723	0.5571
Dataset 4	Phase 1	0.5541	0.5583	0.5462	0.5537	0.5619	0.5607
	Phase 2	0.5702	0.5533	0.5356	0.5770	0.5691	0.5433

TABLE 5.15 – Résultats de cross-validation sans Random avec K=15

5.2.2 K-fold cross-validation avec Random

Les trois tableaux suivants représentent l'évaluation des cinq algorithmes de recommandation sur quatre jeux de données, évalués par la technique de validation croisée K-fold avec *Random* et avec (K=5, 10 et 15) en deux phases sans et avec météo.

Algorithme		H-RN	KNN (K=10)	KNN (K=15)	RF	NN	NB
Dataset 1	Phase 1	0.5934	0.5781	0.5537	0.5947	0.5814	0.5622
	Phase 2	0.6227	0.5844	0.5690	0.6132	0.6122	0.5937
Dataset 2	Phase 1	0.5807	0.5549	0.5380	0.5786	0.5965	0.5851
	Phase 2	0.5995	0.5607	0.5534	0.5955	0.6081	0.6054
Dataset 3	Phase 1	0.6172	0.6033	0.5764	0.6159	0.5840	0.5776
	Phase 2	0.6344	0.6091	0.5871	0.6350	0.6157	0.6108
Dataset 4	Phase 1	0.5906	0.5877	0.5641	0.5904	0.5937	0.5729
	Phase 2	0.5994	0.5963	0.5806	0.5969	0.6069	0.6033

TABLE 5.16 – Résultats de cross-validation avec Random pour K=5

Algorithme		H-RN	KNN (K=10)	KNN (K=15)	RF	NN	NB
Dataset 1	Phase 1	0.5772	0.5600	0.5466	0.5922	0.5561	0.5418
	Phase 2	0.6182	0.5592	0.5476	0.6081	0.6133	0.5968
Dataset 2	Phase 1	0.5540	0.5327	0.5473	0.5545	0.5639	0.5472
	Phase 2	0.5937	0.5733	0.5362	0.5932	0.6072	0.6041
Dataset 3	Phase 1	0.5937	0.5961	0.5606	0.5989	0.5800	0.5869
	Phase 2	0.6049	0.5616	0.5831	0.6071	0.6208	0.6027
Dataset 4	Phase 1	0.5855	0.5633	0.5519	0.5908	0.5784	0.5757
	Phase 2	0.6002	0.5772	0.5644	0.5945	0.6043	0.5965

TABLE 5.17 – Résultats de cross-validation avec Random pour K=10

Algorithme		H-RN	KNN (K=10)	KNN (K=15)	RF	NN	NB
Dataset 1	Phase 1	0.5534	0.5567	0.5372	0.5922	0.5693	0.5437
	Phase 2	0.6237	0.5961	0.5729	0.6209	0.6218	0.6196
Dataset 2	Phase 1	0.5481	0.5580	0.5439	0.5449	0.5551	0.5288
	Phase 2	0.6022	0.5837	0.5610	0.5986	0.6147	0.6111
Dataset 3	Phase 1	0.5790	0.5492	0.5522	0.5733	0.5608	0.5461
	Phase 2	0.6184	0.6033	0.5975	0.5945	0.6004	0.6059
Dataset 4	Phase 1	0.5962	0.5673	0.5497	0.5940	0.5830	0.5537
	Phase 2	0.6265	0.6052	0.5876	0.6222	0.6182	0.6094

TABLE 5.18 – Résultats de cross-validation avec Random pour K=15

Vous pouvez clairement voir que toutes les méthodes d'apprentissage automatique sont en mesure d'améliorer la précision des prévisions à partir de la phase météorologique dans tous les cas, de sorte que le modèle a clairement appris quelque chose d'utile. Par conséquent, nous évaluons chaque algorithme sur des données de test avec les paramètres suivants : K = 5,10 et 15. Sans paramètre aléatoire.

En comparant la meilleure précision pour RF, NB, NN, KNN et H-RN, nous concluons que notre algorithme proposé fonctionne mieux dans cette étude. H-RN fournit des recommandations de haute qualité pour une énorme quantité de données. Par exemple, il atteint la meilleure précision de 62,88 % pour k-fold = 5 (sans para-

mètre aléatoire/avec météo) sur l'ensemble de données 3 et 63,44 % pour k-fold = 5 (avec un paramètre aléatoire/avec météo) sur le même ensemble de données, les mauvaises valeurs sont de l'ordre de k-fold=15 (54,81 %) fournies par la validation croisée de H-RN mais les différences peuvent encore être assez importantes avec KNN-K=15 (51,16 %) sur k-fold= 10.

En comparant les résultats de précision ci-dessus avec les résultats de précision que nous avons calculés auparavant dans la phase de validation fractionnée, nous pouvons voir que les différences sont parfois assez importantes. Par conséquent, la précision des modèles à la fois en validation fractionnée et en validation croisée semble variée considérablement et cela peut être dû au décalage temporel entre l'entraînement et le test, ce qui fait que certains modèles perdent leur capacité à généraliser aussi efficacement dans les données futures. D'après les résultats discutés ci-dessus, la validation croisée k-fold est la meilleure méthode possible chaque fois que l'on veut valider l'accuracy d'un modèle prédictif.

5.3 Tests statistiques

Dans cette sous-section, nous présentons des études statistiques pour évaluer nos expérimentations de recommandation. Les techniques sont bien connues dans les domaines de la recherche d'informations et de l'apprentissage automatique.

La qualité d'un système de recommandation est définie en termes de métriques statistiques, de métriques d'erreur et de métriques de précision. Nous réalisons dans cette partie des tests statistique les plus populaires dans le domaine de la comparaison d'algorithmes de prédiction. (ANalysis Of VAriance) [Kim, 2014] est une technique de test d'hypothèses utilisée pour déterminer si deux populations (ou traitements) ou plus sont statistiquement différentes les unes des autres. Ainsi, la diversité représente la mesure de la qualité de variété entre les éléments de la recommandation [K. Matevz, 2017].

Nous illustrons dans ce qui suit l'utilisation de l'ANOVA à un facteur et de la diversité, sachant que les deux méthodes sont récentes, et très efficaces d'après plusieurs études [K. Matevz, 2017] [Kim, 2014] pour détecter la qualité d'un système statistiquement.

5.3.1 One Way ANOVA

One Way ANOVA est une technique d'évaluation qui fait partie de la famille de la technique normal distrubution Mean [Kim, 2014].

Dans le cas où nous avons seulement deux datasets à comparer, nous devrons calculer *paired t-test* ou *impaired t-test*. Nous choisissons *paired t-test* dans le cas où les deux datasets contenant le même groupe d'individu, sinon nous sommes dans le cas de *impaired t-test*.

Dans le cas de plus de deux datasets, nous trouverons deux techniques, *repeated measures ANOVA* si les datasets contenant le même groupe d'individu (paired), ou *One Way ANOVA* si les individus des jeux de données sont différentes (inpaired). Dans notre cas, nous avons cinq systèmes de recommandation avec des individus différents ce qui nous permet de calculer *One Way ANOVA*.
One Way ANOVA permet de comparer la variance entre les résultats et tirer des conclusions sur les deux hypothèses étudiées :

- Hypothèse (H0) : Il n'y a pas de différence significative dans les résultats obtenus par les cinq systèmes de recommandation.

- Hypothèse (H1) : Il existe une différence significative dans les résultats obtenus par les cinq systèmes de recommandation.

Notre but est de valider l'hypothèse (H1), pour dire que les résultats des cinq bases de données ne sont pas similaires, car si les résultats sont similaires alors on dit que les systèmes choisis pour évaluer nos expérimentations ne sont pas fiables (état H0). Pour cela, nous avons fait une expérimentation pour 10 nouveaux utilisateurs, et nous avons

enregistré les résultats des cinq systèmes de recommandation dans le tableau (5.19).

User	S1	S2	S3	S4	S5
U1	4	3,5	3	4,5	4,5
U2	2,5	2	2,5	2	2
U3	3	2	4	3	3
U4	1	1	2	1	1,5
U5	3,5	3	3,5	3	4
U6	4	5	4	5	5
U7	3	2	2,5	2,5	2,5
U8	3	2,5	3,5	3	3,5
U9	5	3	5	4,5	5
U10	2,5	1,5	2,5	1,5	2

TABLE 5.19 – Matrice d'un exemple numérique de la technique One Way ANOVA

Après avoir appliquer la technique de One Way ANOVA comme illustrée à la table (5.19), nous avons obtenu les résultats qui sont représentés dans la table (5.20) par l'application d'une ANOVA à un facteur en utilisant une valeur de = 0,05 (niveau de signification). Si Fc (value of ratio Fc Between) est supérieure à la valeur critique de Ft, nous rejetons l'hypothèse (H0) et nous validons l'hypothèse (H1). Et si Fc est inférieur à la valeur critique de Ft, nous validons l'hypothèse (H0).

SOURCE	Sum of Squares SS	Degrees of Freedom DF	Mean Square MS	Value of Ratio Fc
Between	14	245	17.5	12.455
Within	37	52	1.405	1.249
TOTAL	51	297	18.905	13.704

TABLE 5.20 – Résultats d'ANOVA à un facteur

Nous devons calculer le (Ft) selon la formule suivante :

$$Ft = F(0.05, SSb, SSw) \tag{5.9}$$

Où SSb représente le "Sum of Squares Between" et SSw représente le "Sum of Squares Within".

À l'aide du tableau de répartition F [1], nous croisons la valeur de colonne SSb de 14 et la valeur de ligne SSw de 37 pour obtenir la valeur critique de Ft comme suit :

Ft = F (0,05, 14, 37) = 2,0921.

Nous voyons clairement que la valeur de Fc (Value of Ratio Fc Between) égal à 12.455, et selon F le tableau de répartition, la valeur critique Ft = 2,0921.

Comme Fc est supérieur à la valeur critique Ft (**Fc > Ft car 12,455 > 2,0921**), alors **l'hypothèse (H1) est validée**, cela signifie que les cinq systèmes sont fiables et donnent de bons résultats, et ils ont la capacité d'évaluer la performance de nos expérimentations.

5.3.2 Diversité

La performance du système de recommandation est mesurée aussi par la diversité des items recommandés avec chaque algorithme. La diversification est un problème qui a été largement étudié dans le domaine de la recherche d'information. Cette mesure représente les différentes interprétations de la recherche de l'utilisateur, et aussi elle permettent de découvrir la qualité de divertissement des items recommandés. La formule suivante représente la formule de la diversité :

$$Ipd(P) = \frac{\sum_{i=1}^{n}(1 - sim(i,j))}{(|P| - 1) * |P|} \tag{5.10}$$

Sachant que :

- P représente le package à recommander pour l'utilisateur,
- Ipd représente la diversité intra colis (intra package diversity),
- N représente la taille du package.

La similarité entre les différentes items à recommander est calculé par la distance euclidienne.

Le tableau suivant contient les résultats obtenus par la technique de la diversité.

1. https ://datascience.eu/fr/mathematiques-et-statistiques/f-tableaux-de-repartition/

Algorithme	Sans Météo	Avec Météo
H-RN	1.208	1.472
KNN (K=5)	0.182	1.130
KNN (K=10)	1.067	1.354
KNN (K=15)	1.206	1.466
Forêt aléatoire	1.205	1.436
Réseau neuronal	0.216	1.149
Naïve-Bayes	1.143	1.380

TABLE 5.21 – Résultats de diversité

Le tableau (5.21) montre la diversité des algorithmes susmentionnés dans les deux cas sans et avec contexte météorologique. Il est clair que H-RN a obtenu les meilleurs résultats en termes de diversité à la fois sans et avec météo. La diversité des KNN (K=15) est également louable. Nous concluons que les résultats de notre algorithme sont très gratifiants.

5.4 Métriques d'erreur

Les métriques d'erreur sont des techniques spécialisées dans le calcul des taux d'erreur dans les systèmes de recommandation, ce qui ressort clairement de son nom, elles sont également utilisées dans la recherche d'informations et bien plus encore pour l'évaluation des moteurs de recherche.

Nous avons calculé deux métriques d'erreur, RMSE et MAE que nous présentons dans les deux sous-sections suivantes.

5.4.1 Racine de l'Erreur Carré Moyenne

RMSE est une technique qui mesure l'amplitude moyenne de l'erreur entre la note réelle qui existe dans la base de données, et la note prédite par le système. RMSE permet de dire si le système donne plusieurs fausses notes, c'est à dire de fausses prédictions, ou il est dans les normes. Si la valeur RMSE d'un système est faible, le système fonctionne bien. Les résultats de la RMSE sont présentés dans le tableau

(5.22).

Algorithme	Dataset 1	Dataset 2	Dataset 3	Dataset 4
H-RN	1.40	1.58	1.40	1.51
KNN (K=5)	1.40	1.59	1.43	1.52
KNN (K=10)	1.39	1.61	1.47	1.53
KNN (K=15)	1.41	1.56	1.49	1.55
Forêt aléatoire	1.44	1.52	1.48	1.54
Réseau neuronal	1.38	1.59	1.42	1.57
Naïve-Bayes	1.41	1.47	1.49	1.59
SVD	1.44	1.57	1.5	1.59

TABLE 5.22 – Résultats de RMSE

5.4.2 Erreur Absolue Moyenne

L'erreur absolue moyenne (MAE) est une mesure des erreurs entre deux listes de mêmes tailles et de mêmes types, en général la MAE sert à calculer les moyennes de différences entre deux listes. Nous avons choisi MAE pour vérifier le taux d'erreur entre les notes réelles et les notes récupérées par les cinq systèmes. Les résultats de la MAE sont présentés dans le tableau (5.23)

Algorithme	Dataset 1	Dataset 2	Dataset 3	Dataset 4
H-RN	0.000097	0.000108	0.000101	0.000094
KNN (K=5)	0.000098	0.000096	0.000092	0.000103
KNN (K=10)	0.000099	0.000108	0.000103	0.000097
KNN (K=15)	0.000103	0.000098	0.000095	0.000101
Forêt aléatoire	0.000109	0.000125	0.000117	0.000098
Réseau neuronal	0.000125	0.000117	0.00012	0.000119
Naïve-Bayes	0.000098	0.000098	0.000109	0.000097
SVD	0.00834	0.010523	0.0113	0.00935

TABLE 5.23 – Résultats de MAE

Chaque cellule ci-dessus (tableaux 5.22 et 5.23) montre le test d'erreur de chaque algorithme étudié. MAE et RMSE expriment l'erreur de prédiction moyenne du modèle

en unités de la variable d'intérêt. Les deux mesures sont des scores orientés négativement, ce qui signifie que les valeurs inférieures sont les meilleures. H-RN a donné les meilleurs résultats pour les bases de données 3 et 4, et aussi il a donné de bons résultats pour les BDD 1 et 2. NN a donné le meilleur résultat pour la BDD 1 et NB a dominé les résultats de BDD 2 par un taux de 1.47.

Le classificateur H-RN reprend la tête du MAE dans les ensembles de données 1 et 4. Le MAE élevé se produit sur NN et SVD avec 0,000125 et 0,010523 respectivement. H-RN se place en 2ème et en 3ème position dans les jeux de données 3 et 2 respectivement après KNN et NB. En résumé, à partir de l'analyse ci-dessus, nous pouvons constater que notre proposition peut certainement améliorer les performances de prédiction et de recommandation et diminuer la valeur de RMSE et MAE.

5.5 Étude de complexité

Le but de la complexité d'après [S. Maguire, 2006] est de mesurer la qualité et la rapidité d'un algorithme et de permettre une comparaison directe entre plusieurs algorithmes en fonction du temps d'exécution ou du nombre d'instructions de chaque algorithme. Dans ce qui suit, nous allons calculer la complexité de chaque algorithme utilisé dans notre étude. Le tableau (5.24) représente la complexité de H-RN, KNN, RF, NB et NN.

Algorithme	H-RN	KNN	RF	NB	NN
Complexité	$O(n^3 + log(n))$	$O(\log(n^2))$	$O(n^2 + 2n)$	$O(n^3 + log(2n))$	$O(n^3 + 2n)$

TABLE 5.24 – Complexité des algorithmes de la deuxième approche

La complexité de H-RN donne de bons résultats pour $N > 10000$, tandis que la complexité de KNN est faible pour $N < 10000$, donc dans le cas où N est supérieur à 10 000, KNN prend une longue durée pour récupérer les résultats, sachant que la taille des jeux de données 1,2, et 4 de notre étude est supérieure à 10000. La complexité de RF est inférieure à notre algorithme, mais avec une légère différence, comme pour NB, il a la même complexité avec H-RN. Par conséquent, nous pouvons conclure que notre algorithme peut être utilisé en pratique car il est de complexité polynomiale, efficace et très rapide.

6 Conclusion

Nous avons présenté dans ce dernier chapitre notre approche proposée pour la recommandation des items touristiques. Cette dernière est créée pour recommander les k-items qui répondent aux besoins de l'utilisateur. En d'autres termes, l'enjeu était de prédire les recommandations les plus appropriées pour visiter un service touristique approprié. Les algorithmes d'apprentissage automatique peuvent être utiles pour faire des prédictions pour les données.

L'utilisation des techniques d'apprentissages automatiques permet de réaliser et d'améliorer un système de recommandation touristique sensible au contexte en analysant non seulement les préférences, le profil et le contexte de l'utilisateur, mais également le contexte de la destination souhaitée. Le filtrage collaboratif basé sur un modèle et le filtrage collaboratif basé sur la mémoire sont intégrés dans le modèle de recommandation proposé qui est basé sur une large gamme d'algorithmes de machine learning.

La combinaison des deux approches a donné les meilleurs résultats dans notre travail. Leur fusion était importante car le premier modèle est appliqué à une prédiction et l'autre modèle est appliqué à une phase de recommandation. La validation croisée est le meilleur moyen de tirer pleinement parti des données sans perte d'informations lors de la phase d'apprentissage.

L'objectif de cette recherche est d'améliorer la précision des prédictions. Une analyse de différents algorithmes de ML tels que Naïve Bayes, Random Forest, Neural Network et KNN a été réalisée, ainsi qu'une étude détaillée de l'algorithme H-RN proposé a été réalisée pour savoir si notre algorithme hybride est plus performant que les autres algorithmes d'apprentissage automatique existants. Pour évaluer notre étude, nous avons mesuré la précision des recommandations grâce à une série d'expériences avec des ensembles de données réels. H-RN fournit des recommandations de haute qualité et obtient de meilleurs résultats dans cette étude.

Conclusion Générale et Perspectives

Dans cette thèse, nous nous sommes intéressé à l'optimisation du problème du plus court chemin ainsi qu'aux systèmes de recommandation des endroits touristiques à l'aide des techniques de machine learning. Notre but était de concevoir un système capable de satisfaire l'utilisateur selon son profil, ses préférences et son contexte en temps réel. Dans notre proposition, le contexte a été représenté par la situation météorologique et par le concept de localisation.

Après avoir présenté le contexte, la problématique et les verrous sous-tendus, cette thèse a décrit nos motivations et contributions qui se situe à l'intersection du paradigme d'optimisation combinatoire et des systèmes de recommandation. Dans la première partie en état de l'art, nous avons présenté (i) quelques notions sur l'optimisation à savoir : la théorie des graphes et quelques techniques d'optimisation ainsi que (ii) des généralités sur les systèmes de recommandation telles que : les différentes approches de recommandation, quelques algorithmes et métriques d'évaluation utilisés dans la recommandation, et la notion de profil, préférences et contexte. Ensuite, nous avons illustré les travaux connexes qui ont été réalisés dans la littératures relatifs à l'optimisation des chemins les plus courts et les systèmes de recommandation sensibles au contexte avec une étude comparative des recherches existantes.

Ce manuscrit a détaillé dans la deuxième partie nos deux contributions proposées dans le cadre de cette thèse. La première contribution consiste à un nouvel algorithme génétique hybride nommé IOGA qui permet de trouver un ou k chemins ayant le coût minimal entre deux sommets dans un grand graphe pondéré orienté et non orienté. IOGA est une hybridation de l'algorithme exact (Dijkstra) avec l'algorithme métaheuristique (Genetic Algorithm). Cet algorithme a utilisé des stratégies qui se sont avérées efficaces pour résoudre le problème des plus courts chemins, ainsi il a permis d'améliorer les performances de calcul de notre problème dans un réseau routier. Les tests ont été effectués à l'aide de réseaux générés aléatoirement. L'étude expérimentale a démontré l'efficacité de L'IOGA en termes de temps d'exécution et de qualité des résultats. Les résultats empiriques obtenus montrent que l'IOGA est meilleur que certaines techniques existantes en termes de solution optimale à chaque génération ainsi que le nombre des chemins retournés, la complexité et le temps d'exécution.

La deuxième contribution consiste à utiliser des techniques et des algorithmes de machine learning qui peuvent prédire et proposer des services touristiques pertinents (k-

items) aux utilisateurs selon leurs intérêts, leurs besoins, leurs goûts et leurs contextes. L'objectif principal est de fournir automatiquement des recommandations personnalisées aux demandeurs de services en tenant compte à la fois de leurs profils, leurs préférences et de leurs contextes implicites/explicites. Un nouvel algorithme efficace et intelligent nommé H-RN a été proposé, il hybride à la fois deux algorithmes d'apprentissage automatique les plus connus, à savoir Random Forest et Naîve Bayes, avec un filtrage collaboratif (technique basée sur le modèle et basée sur la mémoire).

H-RN a été appliqué avec quatre algorithmes de machine learning (RF, NB, KNN, NN) sur quatre différentes bases de données. Les résultats de notre approche montrent l'efficacité de l'utilisation de la phase météo dans la majorité des algorithmes utilisés. H-RN donne les meilleurs résultat d'après les mesures d'évaluation dans la technique de split validation et aussi dans la technique de cross validation. Le résultats du rappel, de la précision, de l'exactitude, de la F-mesure et du taux moyen, ainsi qu'un ensemble de tests statistiques (One-way ANOVA, Diversity) et des mesures d'erreur (RMSE, MAE) ont montré l'amélioration de la précision de prédiction de notre algorithme par rapport aux approches de base dans divers contextes. Le nombre important des expérimentations, ainsi que le choix des algorithmes à hybrider sont des facteurs importants pour avoir de bons résultats.

-
-
- Application de l'IOGA à une étude de cas réel de trafic routier prenant en compte diverses contraintes réelles telles que le problème des feux tricolores ou la détection des travaux ou accidents qui bloquent les routes.

Dans les travaux futurs, pour la première approche, Nous prévoyons d'appliquer IOGA sur d'autres jeux de données et comparer nos résultats avec d'autres approches. Et l'application d'IOGA sur le problème du voyageur de commerce ou/et du Knapsack. Aussi nous avons comme perspective l'hybridation d'IOGA avec d'autres techniques comme la programmation par contraintes. Et aussi d'explorer la possibilité d'étendre les différentes stratégies de recherche heuristique aux réseaux dynamiques et/ou stochastiques. Nous avons aussi des perspectives a long terme qui peuvent être utilisé comme des thèmes de thèse au future, comme l'application de l'IOGA à une étude de cas réel de trafic routier prenant en compte diverses contraintes réelles telles que le

problème des feux tricolores ou la détection des travaux ou accidents qui bloquent les routes.

Et pour la deuxième contribution, nous prévoyons d'améliorer la précision des recommandations en faisant varier le k item ou en combinant notre algorithme avec des algorithmes de clustering (X-means et Fuzzy C-means) pour trouver le meilleur modèle de prédiction. De plus, nous envisageons d'étendre nos expériences avec d'autres ensembles de données publics, d'examiner une analyse plus approfondie et d'améliorer l'évolutivité et l'efficacité de nos contributions. Pour des recherches supplémentaires, extraire d'autres caractéristiques comme les personnes accompagnantes (enfants, épouse, amis, etc.) pourra améliorer la qualité de la recommandation, de plus, en ajoutant des critères supplémentaires dans la compilation de l'algorithme, permettra également d'obtenir de meilleurs résultats. Pour améliorer encore les résultats, une technique d'apprentissage en profondeur peut être envisagée, et enfin, d'autres métriques d'évaluation pourraient également être utilisées telles que les tests non paramétriques et les tests bayésiens pour obtenir une perspective complète de la comparaison des résultats des différents algorithmes. Aussi nous espérons comparé nos résultats avec d'autre algorithme de recommandation comme l'algorithme gradient boosting.

Bibliographie

[A. Gillet, 2011] A. Gillet, Y. Brostaux, e. a. (2011). Principaux modèles utilisés en régression logistique. *Biotechnology, Agronomy and Society and Environment 15(3)*. 53

[A. J. Kulkarni, 2017] A. J. Kulkarni, Ganesh Krishnasamy, e. a. (September 2017). *Introduction to Optimization*. Cohort Intelligence : A Socio-inspired Optimization Method (pp.1-7). 13

[A. K. Guruji, 2016] A. K. Guruji, Himansh Agarwal, D. K. P. (2016). Time-efficient a* algorithm for robot path planning. 24

[A. K. Jeewajee, 2020] A. K. Jeewajee, L. P. K. (2020). Adversarially-learned inference via an ensemble of discrete undirected graphical models. 18

[A. Kenaan, 2020] A. Kenaan, K. Benabdeslem, e. a. (2020). Approches hybrides pour la recommandation dans le domaine du pneumatique. *Université Lyon 1, 43 Boulevard du 11 novembre 1918, Villeurbanne cedex 69622, Lizeo IT, 42 Quai Rambaud, 69002 Lyon*. 47

[A. Kulkarni, 2019] A. Kulkarni, R. M. Samant, P. B. A. P. (2019). A machine learning approach to building a tourism recommendation system using sentiment analysis. 42, 52

[A. Odic, 2013] A. Odic, A. K. (January 2013). Predicting and detecting the relevant contextual information in a movie-recommender system. 56

[A. Roshchina, 2015] A. Roshchina, J. Cardiff, P. R. (2015). Twin : Personality-based intelligent recommender system. 110

[A. Umanetsa, 2014] A. Umanetsa, Artur Ferreiraa, N. L. (2014). Guideme - a tourist guide with a recommender system and social interaction. 42

[A. Yang, 2015] A. Yang, Dingqi, Z. D. (2015). Modeling user activity preference by leveraging user spatial temporal characteristics in lbsns. *IEEE Transactions on Systems, Man, and Cybernetics : Systems*, 45(1) :129–142. 112

[B. Mobasher, 2004] B. Mobasher, X. Jin, Y. Z. (2004). Semantically enhanced collaborative filtering on the we. *In Proceedings of the First EuropeanWeb Mining Forum – EWMF 2003, pages 57–76. Springer, 2004.* 46

[B. S. Abdualgalil, 2020] B. S. Abdualgalil, S. A. (2020). Tourist prediction using machine learning algorithms. 50

[Barbosa, 2014] Barbosa, A. D. (December 10, 2014). Privacy-enabled scalable recommender systems. 57

[Bellman, 1958] Bellman (1958). On a routing problem. 15, 21

[BenFadhel, 2019] BenFadhel, K. (2019). Amplification d'arbres de régression compatibles avec l'encodage de la sortie, application à la reconnaissance des images de chiffres manuscrits. *Université de Laval, Québec, Canada.* 53

[Benouaret, 2015] Benouaret, I. (2015). Un système de recommandation sensible au contexte pour la visite de musée. 59

[Benouaret, 2017] Benouaret, I. (2017). Un système de recommandation contextuel et composite pour la visite personnalisée de sites culturels. 39, 40, 49

[BenTicha, 2015] BenTicha, S. (novembre 2015). Recommandation personnalisée hybride. 44

[Bernard, 1959] Bernard, F. W. (1959). Transitivité et connexité. *Comptes rendus hebdomadaires des séances de l'Académie des sciences.* 23

[Boisson, 2008] Boisson, J. (2008). Modélisation et résolution par métaheuristiques coopératives : de l'atome à la séquence protéique. *Laboratoire d'Informatique Fondamentale de Lille — UMR CNRS 8022.* 31

[Botchkarev, 2019] Botchkarev, A. (2019). Performance metrics (error measures) in machine learning regression, forecasting and prognostics : Properties and typology. *Interdisciplinary Journal of Information, Knowledge, and Management, 2019, 14, 45-79.* 125

[Bouyer, 2020] Bouyer, J. (2020). Régression logistique - modélisation des variables quantitatives. *Université Paris Saclay, France.* 53

[C. Cerrone, 2017] C. Cerrone, R. Cerulli, e. a. (2017). Carousel greedy : A generalized greedy algorithm with applications in optimization. *Computers Operations Research 85.* 27

[C. G. Rodriguez, 2012] C. G. Rodriguez, D. F.-G. (April 23 - 27 2012). Dependency parsing with undirected graphs. 18

[Caussinus, 1993] Caussinus, H. (1993). Modèles probabilistes et analyse des données multidimensionnelles. quelques réflexions méthodologiques et applications. 44

[C.S. Namahoot, 2015] C.S. Namahoot, M. Bruckner, N. P. (2015). Context-aware tourism recommender system using temporal ontology and naïve bayes. *In Recent Advances in Information and Communication Technology.* 72, 74

[D. Goldberg, 1992] D. Goldberg, D. Nichols, B. O. D. T. (1992). Using collaborative filtering to weave an information tapestry. 38, 66

[D. Salber, 1999] D. Salber, Dey, e. a. (1999). The context toolkit : Aiding the development of context-enabled applications. 59

[D. Werner, 2014] D. Werner, Christophe Cruz, A. B. (January 2014). Evaluation de la pertinence dans un système de recommandation sémantique de nouvelles économiques. 55

[Demsar, 2006] Demsar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research 7, 2006, pp 1-30.* 125

[Dijkstra, 1959] Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. 21

[F. Celli, 2014] F. Celli, e. a. (2014). The workshop on computational personality recognition. 110

[F. Duchon, 2014] F. Duchon, Andrej Babinec, e. a. (2014). Path planning with modified a star algorithm for a mobile robot. 68

[F. Glover, 1986] F. Glover, M. L. (1986). rabu search. *Book in Informs Journal on Computing.* 29

[F. Simsir, 2019] F. Simsir, D. E. (2019). A metaheuristic solution approach to capacitied vehicle routing and network optimization. 65, 68, 81

[Ford, 1956] Ford, J. (1956). Network flow theory. 21

[G. Adomavicius, 2005] G. Adomavicius, A. T. (2005). Toward the next generation of recommender systems : A survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering.* 43

[Gardeux, 2011] Gardeux, V. (2011). Conception d'heuristiques d'optimisation pour les problèmes de grande dimension. application à l'analyse de données de puces à adn. 29

[Grabener, 2011] Grabener, T. (2011). Calcul d'itineraire multimodal et multiobjectif en milieu urbain. 22

[Gras, 2018] Gras, B. (2018). Les oubliés de la recommandation sociale. 48

[Guillot, 2020] Guillot, M. (2020). Le problème du plus court chemin stochastique et ses variantes : fondements et applications à l'optimisation de stratégie dans le sport. 20

[H. Bast, 2015] H. Bast, Daniel Delling, e. a. (April 17, 2015). Route planning in transportation networks. 64

[H. Bouhissia, 2021] H. Bouhissia, Micipsa Adel, e. a. (March 24–26 2021). Towards an efficient knowledge-based recommendation system. 41

[H. Kashif, 2019] H. Kashif, Mohd Najib Mohd Salleh, e. a. (2019). Metaheuristic research : a comprehensive survey. 24

[H. S. Jigarkumar, 2020] H. S. Jigarkumar, R. H. J. (February 2020). Hybrid user clustering-based travel planning system for personalized point-of-interest recommendation. 69, 74

[H. Sachs, 1988] H. Sachs, M. S. (1988). An historical note : Euler's konigsberg letters. *THE OPEN UNIVERSITY MILTON KEYNES, ENGLAND.* 16

[H. Yusnita, 2020] H. Yusnita, M. Zarlis, S. (2020). Performance of a star with dynamic programming algorithms in determining the shortest route. *In Proceedings of the International Conference on Culture Heritage, Education, Sustainable Tourism, and Innovation Technologies (CESIT 2020), pages 536-542.* 24

[Hamed, 2010] Hamed, A. Y. (2010). A genetic algorithm for finding the k shortest paths in a network. *Egyptian Informatics Journal, Volume 11, Issue 2, December 2010, Pages 75-79.* 24

[Haydar, 2014] Haydar, C. A. (2014). Les systèmes de recommandation à base de confiance. 54

[Hosseinabadi, 2018] Hosseinabadi, J. Vahidi, e. a. (2018). Extended genetic algorithm for solving open-shop scheduling problem. 24, 82

[Huang, 2009] Huang, Bian, L. A. (2009). Bayesian network and analytic hierarchy process based personalized recommendations for tourist attractions over the internet. *Expert Systems with Applications*. 104

[I. Maatouk, 2017] I. Maatouk, Iman Jarkass, E. C. N. C. (2017). Preventive maintenance optimization and comparison of genetic algorithm models in a series–parallel multi-state system. 20, 68

[J. A. Konstan, 2012] J. A. Konstan, J. R. (2012). Recommender systems : from algorithms to user experience. 38

[J. Arshad, 2020] J. Arshad, Muhammad Tauhidur Rahman, e. a. (March 2020). Intelligent intersection control for delay optimization : Using meta-heuristic search algorithms. 64, 65, 68

[J. Beel, 2016] J. Beel, Bela Gipp, e. a. (2016). Research-paper recommender systems : a literature survey. 37

[J. Deneubourg, 1990] J. Deneubourg, S. Aron, S. G. J. P. (1990). The self-organizing exploratory pattern of the argentine ant. *Journal of Insect Behavior*. 29

[J. L. Gross, 2013] J. L. Gross, Jay Yellen, P. Z. (2013). *History of Graph Theory*. Handbook of Graph Theory CRC Press. 16

[J. Lee, 2012] J. Lee, J. Y. (2012). A fast and scalable re-routing algorithm based on shortest path and genetic algorithms. 66, 67, 81, 82

[J. Nator, 2020] J. Nator, J. E. B. M. (2020). An intersection traffic signal controller optimized by a genetic algorithm. 64, 65, 68

[k. Domokos, 2019] k. Domokos, Bálint Daróczy, e. a. (2019). Session recommendation via recurrent neural networks over fisher embedding vectors. 52

[K. Gautam, 2017] K. Gautam, D. Fabio, e. a. (2017). Netflix : An in-depth study of their proactive adaptive strategies to drive growth and deal with issues of net-neutrality digital equity. *IRA International Journal of Management Social Sciences*. 37

[K. JunWoo, 2018] K. JunWoo, S. K. K. (2018). Genetic algorithms for solving shortest path problem in maze-type network with precedence constraints. 66

[K. L. Umesh, 2021] K. L. Umesh, Sarita Simaiya, e. a. (2021). Hybrid weighted random forests method for prediction classification of online buying customers. 69, 74

[K. Lakiotaki, 2010] K. Lakiotaki, Nikolaos F. Matsatsinis, A. T. (2010). Multi-criteria user modeling in recommender systems. 58

[K. Matevz, 2017] K. Matevz, P. T. (2017). Diversity in recommender systems – a survey. *Knowledge-Based Systems Volume 123, 2017, pp. 154-162.* 143

[Kacem, 2012] Kacem, F. (2012). Algorithmes exacts et approchés pour des problèmes d'ordonnancement et de placement. *Laboratoire IBISC EA 4526 - Université d'Évry-Val d'Essonne.* 24

[Ketfi, 2016] Ketfi, A. C. (2016). Sur certaines méthodes d'optimisation globale basées sur l'introduction de fonctions auxiliaires. 13

[Kim, 2014] Kim, H. (2014). Analysis of variance (anova) comparing means of more than two groups. *Restorative Dentistry Endodontics, vol. 39, no. 1, The Korean Academy of Conservative Dentistry, 2014, p. 74.* 143, 144

[L. Georgiadis, 2014] L. Georgiadis, Giuseppe F. Italiano, e. a. (September 2014). 2-vertex connectivity in directed graphs. 17

[L. Shuxian, 2019] L. Shuxian, F. S. (2019). Design and implementation of movie recommendation system based on naive bayes. 49

[L. Xing, 2019] L. Xing, Y. L. (2019). Revised floyd-warshall algorithm to find optimal path in similarityweight network and its application in the analysis of global value chain. 23

[Lang, 1995] Lang, K. (1995). Newsweeder : Learning to filter netnews. *In in Proceedings of the 12th International Machine Learning Conference (ML95, 1995).* 40

[Lebbah, 2015] Lebbah, F. Z. (2015). Programmation par contrainres appliquée à des problèmes issus des finances. *Laboratoire d'informatique et des technologies de l'information d'Oran (LITIO).* 27, 31

[Lemdani, 2016] Lemdani, R. (2016). Système hybride d'adaptation dans les systèmes de recommandation. *Université Paris Saclay (COmUE), 2016. Français. ffNNT : 2016 SACLC050ff. fftel-01371650.* 46

[M. Balabanovic, 1997] M. Balabanovic, Y. S. (1997). content-based, collaborative recommendation. 38

[M. Changxi, 2018] M. Changxi, Ruichun He, W. Z. (2018). Path optimization of taxi carpooling. 20, 68

[M. Debnath, 2016] M. Debnath, Tripathi, P. E. R. (2016). Preference-aware successive poi recommendation with spatial and temporal influence. *Lecture Notes in Computer Science*. 104

[M. F. Khalfi, 2014] M. F. Khalfi, S. M. B. (June 2014). Systèmes d'information pervasifs : Architecture et challenges. 59, 61

[M. Kirsch, 2007] M. Kirsch, Marlène Villanova-Oliver, e. a. (2007). Un mécanisme d'adaptation guidé par le contexte en utilisant une représentation par objets. 59

[M. Mohri, 2018] M. Mohri, Rostamizadeh, A. T. A. (2018). Foundations of machine learning. *MIT press*. 36

[M. Nasr, 2020] M. Nasr, O. F. (2020). Benchmarking meta-heuristic optimization. *International Journal of Advanced Networking and Applications - IJANA*. 12

[M. Pazzani, 1997] M. Pazzani, D. B. (1997). Learning and revising user profiles : The identification of interesting web sites. *Machine Learning*. 40

[M. Y. Hayi, 2022] M. Y. Hayi, Z. Chouiref, H. M. (2022). Towards intelligent road traffic management over a weighted large graphs hybrid metaheuristic-based approach. *Journal of Cases on Information Technology*. 4, 6, 9, 79

[M. Y. Hayi, 2020] M. Y. Hayi, Z. C. (2020). An improved optimization algorithm to find multiple shortest paths over large graph. *IEEE/ Second International Conference on Embedded Distributed Systems (EDiS),university of Oran, Algeria*. 9

[M. Y. Hayi, 2021] M. Y. Hayi, Z. C. (2021). An efficient hybrid meta-heuristic approach for solving the k-shortest paths problem over weighted large graphs. *Springer/ 4th International Conference on Artificial Intelligence in Renewable Energetic Systems (IC-AIRES), university of Tipasa, Algeria*. 9

[Maatallah, 2016] Maatallah, M. (2016). Une technique hybride pour les systèmes de recommandation. 43

[Manseur, 2017] Manseur, F. (2017). Algorithmes de guidage optimal des usagers dans les réseaux de transport. 20

[Michailidis, 2017] Michailidis, M. (2017). Investigating machine learning methods in recommender systems improving prediction for the top k items. 55

[N. L. Biggs, 1998] N. L. Biggs, E. K. Lloyd, R. J. W. (1998). Graph theory 1736-1936. 16

[N. Shanmugasundaram, 2019] N. Shanmugasundaram, K. Sushita, S. P. K. E. G. (2019). Genetic algorithm-based road network design for optimising the vehicle travel distance. 65

[O. Ugurlu, 2018] O. Ugurlu, D. T. (May 2018). *A Hybrid Genetic Algorithm for Minimum Weight Dominating Set Problem.* Recent Developments and the New Direction in Soft-Computing Foundations and Applications (pp.137-148). 65, 68

[P. Melville, 2002] P. Melville, J. Mooney, e. a. (2002). Content-boosted collaborative filtering for improved recommendations. *Eighteenth National.* 47

[P. Resnick, 1997] P. Resnick, H. V. (1997). Recommender systems. 38

[P. Sanchez, 2021] P. Sanchez, Bellogin, A. (2021). Point-of-interest recommender systems : A survey from an experimental perspective. *ACM Comput.* 104

[Pape, 1974] Pape, U. (1974). Implementation and efficiency of moore-algorithms for the shortest route problem. *Mathematical Programming.* 14, 21

[Polyak, 2007] Polyak, B. T. (September 2007). Newton's method and its use in optimization. 15

[Powers, 2011] Powers, D. (2011). Evaluation : from precision, recall and f-measure to roc, informedness, markedness and correlation. *Journal of Machine Learning Technologies 2, 2011, no. 1, pp. 37-63.* 125

[R. Burke, 2011] R. Burke, Alexander Felfernig, e. a. (2011). Recommender systems : An overview. 37

[R. Fan, 2008] R. Fan, K. Chang, e. a. (2008). Liblinear : A library for large linear classification. *Journal of Machine Learning Research 9 2008, 1871-1874.* 138

[R. Khorsand, 2020] R. Khorsand, Majid Rafiee, V. K. (2020). Insights into tripadvisor's online reviews : The case of tehran's hotels. 69, 74

[Reddy, 2013] Reddy, H. (December 13, 2013). Path finding - dijkstra's and a* algorithm's. 67, 68

[S. Basodi, 2020] S. Basodi, J.Chunyan, e. a. (2020). Gradient amplification : An efficient way to train deep neural networks. *Big Data Mining and Analytics.* 53

[S. Broumi, 2020] S. Broumi, Shio Gai Quek, e. a. (2020). *Finding the Shortest Path With Neutrosophic Theory : A Tool for Network Optimization.* Neutrosophic Graph Theory and Algorithms, DOI : 10.4018/978-1-7998-1313-2.ch001. 65, 68

[S. C. Abraham, 2015] S. C. Abraham, G. D. S. (2015). Shortest path computation in large graphs using bidirectional strategy and genetic algorithms. 66

[S. Kirkpatrick, 1983] S. Kirkpatrick, C. D. Gelatt, J. M. P. V. (1983). Optimization by simulated annealing. *Science.* 27

[S. Kulkarni, 2020] S. Kulkarni, Rodd, S. (2020). Context aware recommendation systems : A review of the state of the art techniques. *Computer Science Review.* 36, 104

[S. Maguire, 2006] S. Maguire, Bill McKelvey, e. a. (2006). complexity science and organization studies. 15, 98, 149

[S. Pettie, 2003] S. Pettie, Vijaya Ramachandran, S. S. (2003). Experimental evaluation of a new shortest path algorithm. 66

[S. Zhang, 2017] S. Zhang, L. Xuelong, e. a. (2017). Learning k for knn classification. *ACM Transactions on Intelligent Systems and Technology 8(3) :1-19.* 52

[Singh, 2018] Singh, S. Sharma, R. S. D. H. (2018). Towards use of dijkstra algorithm for optimal navigation of an unmanned surface vehicle in a real-time marine environment with results from artificial potential field. 22, 67, 68, 83

[Sommer, 2013] Sommer, C. (2013). Shortest-path queries in static networks. 66

[T. Abeywickrama, 2016] T. Abeywickrama, Muhammad Aamir Cheema, D. T. (2016). k-nearest neighbors on road networks : A journey in experimentation and in-memory implementation. 64

[T. Chai, 2014] T. Chai, R. R. D. (2014). Root mean square error (rmse) or mean absolute error (mae) ? – arguments against avoiding rmse in the literature. 56

[T. Ucar, 2019] T. Ucar, A. Karahoca, D. K. (2019). Ntrs : A new travel recommendation system framework by hybrid data mining. *International Journal of Mechanical Engineering and Technology, 10(1) : 935-946.* 72, 74

[Tadlaoui, 2018] Tadlaoui, M. (2018). Système de recommandation de ressources pédagogiques fondé sur les liens sociaux : formalisation et évaluation. 55

[U. Shardanand, 1995] U. Shardanand, M. P. (1995). Social information filtering : algorithms for automating “word of mouth”. 38

[V. K. Muneer, 2020] V. K. Muneer, K. P. M. B. (2020). The evolution of travel recommender systems : A comprehensive review. xiii, 70, 71, 74

[V. Vellaichamy, 2017] V. Vellaichamy, V. K. (2017). Hybrid collaborative movie recommender system using clustering and bat optimization. 70, 74

[Villalobos, 2014] Villalobos, O. B. (2014). Content-based instruction : A relevant approach of language teaching. 40

[W. Shafqat, 2020] W. Shafqat, Y. B. (2020). A context-aware location recommendation system for tourists using hierarchical lstm model sustainability. *12 : 4107 http ://dx.doi.org/10.3390/su12104107.* 72, 74

[Wang, 2021] Wang, H. (2021). Research on personalized recommendation method of popular tourist attractions routes based on machine learning. *International Journal of Information and Communication Technology.* 104

[X. Zhou, 2020] X. Zhou, Su, M. L. Z. H. Y. S. B. F. G. (2020). Smart tour route planning algorithm based on naïve bayes interest data mining machine learning. *ISPRS Int.* 104

[Y. Cai, 2010] Y. Cai, D. Ji, D. C. (2010). A knn research paper classification method based on shared nearest neighbor. *Proceedings of NTCIR-8 Workshop Meeting, June 15–18, 2010, Tokyo, Japan.* 52

[Y. Xu, 2018] Y. Xu, R. G. (2018). On splitting training and validation set : A comparative study of cross-validation, bootstrap and systematic sampling for estimating the generalization performance of supervised learning. *Journal of Analysis and Testing, 2018, 2(3), pp. 249-262.* 126, 127

[Z. Chouiref, 2022] Z. Chouiref, M. Y. H. (2022). Toward preference and context-aware hybrid tourist recommender system based on machine learning techniques. *Revue d'Intelligence Artificielle.* 4, 6, 9

[Z. Fayyaz, 2020] Z. Fayyaz, Mahsa Ebrahimian, e. a. (2020). Recommendation systems : Algorithms, challenges, metrics, and business opportunities. 41

[Zhenyu, 2012] Zhenyu, Jiawei Yao, R. Y. e. a. (November 2012). Product recommendation based on search keywords. 41